VISUAL BASIC NET FORM

LUIS MINGUILLÓN PASCUAL

Visual Basic NET
FORM

Luis Minguillón Pascual

COPYRIGHT

La licencia de este libro electrónico es para uso personal. Por lo tanto no puedes revenderlo a otras personas.

Gracias por respetar los derechos de autor.

©Luis Minguillón Pascual
27/09/2025

Prefacio

Visual Basic .NET (VB.NET) es un lenguaje de programación desarrollado por Microsoft que permite crear aplicaciones tanto de consola como de interfaz gráfica de usuario (GUI) de manera eficiente y con un enfoque altamente visual. Su principal fortaleza radica en la facilidad para desarrollar aplicaciones sobre la plataforma .NET, aprovechando sus bibliotecas y componentes predefinidos que simplifican tareas complejas, como la manipulación de archivos, la conexión a bases de datos o el diseño de formularios interactivos. VB.NET combina la simplicidad del lenguaje Visual Basic clásico con la potencia y flexibilidad de la programación orientada a objetos moderna, lo que lo convierte en una herramienta ideal tanto para principiantes como para programadores experimentados que buscan desarrollar soluciones rápidas y robustas para entornos Windows.

El propósito de este libro es guiar al lector paso a paso en el aprendizaje de Visual Basic .NET enfocado en formularios, proporcionando ejemplos claros, ejercicios prácticos y explicaciones detalladas para que cualquier persona pueda crear aplicaciones funcionales y eficientes. Cada capítulo está diseñado para desarrollar habilidades progresivamente, desde la comprensión básica del lenguaje hasta la creación de proyectos completos que integren distintos componentes y funcionalidades.

Quiero aprovechar este espacio para expresar mi más profundo agradecimiento a mi esposa, **Marisa**, cuyo apoyo incondicional y constante motivación hicieron posible la realización de este libro. Su paciencia, consejos y entusiasmo fueron una fuente de inspiración que me acompañó en cada página escrita y en cada concepto desarrollado. Sin su ayuda, este proyecto no habría alcanzado la forma en que hoy se presenta.

Espero que este libro sea una guía útil, práctica y motivadora para todos aquellos que deseen aprender y dominar Visual Basic .NET en el desarrollo de aplicaciones de escritorio.

Luis Minguillón Pascual

ÍNDICE

Parte I – Fundamentos de la Programación en VB.NET

Capítulo 1. Introducción a Visual Basic .NET

- 1.1. Historia y evolución de Visual Basic
- 1.2. El entorno .NET Framework y .NET Core
- 1.3. Instalación y configuración de Visual Studio
- 1.4. Estructura de un proyecto de Windows Forms

Capítulo 2. Conceptos Básicos de Programación

- 2.1. Estructura de un programa en VB.NET
    - Elementos principales: Declaración de clase, Constructor (Sub New), Procedimientos y funciones, Eventos
- 2.2. Sentencias, bloques e indentación
- 2.3. Uso de comentarios en el código

Capítulo 3. Variables y Tipos de Datos

- 3.1. Declaración y asignación de variables
    - Reglas para nombres de variables
- 3.2. Tipos de datos primitivos y compuestos
- 3.3. Variables estáticas y dinámicas
- 3.4. Conversión y validación de tipos
- 3.5. Constantes y enumeraciones

Capítulo 4. Operadores y Expresiones

- 4.1. Operadores aritméticos, lógicos y relacionales
    - Operadores aritméticos
    - Operadores relacionales
    - Operadores lógicos (And, Or, Not, AndAlso, OrElse)
- 4.2. Operadores de asignación
- 4.3. Operadores de concatenación de cadenas
- 4.4. Expresiones condicionales
    - Estructura If básica
    - Operador ternario If (inline)

Capítulo 5. Estructuras de Control

- 5.1. Condicionales: If, ElseIf, Select Case
- 5.2. Bucles: For, While, Do Until / Do While
- 5.3. Estructuras anidadas
- 5.4. Uso de Exit, Continue y Return

Parte II – Programación Orientada a Objetos

Capítulo 6. Introducción a la POO en VB.NET

- 6.1. Concepto de Clases y Objetos
- 6.2. Constructores y Destructores
- 6.3. Encapsulación, Herencia y Polimorfismo
- 6.4. Sobrecarga de métodos

Capítulo 7. Propiedades, Métodos y Eventos

- 7.1. Creación de Propiedades en Clases
  - Uso de bloques Get y Set para encapsular la lógica de lectura y escritura
- 7.2. Métodos Públicos, Privados y Compartidos (Shared)
  - Métodos Públicos y Privados
  - Métodos Compartidos (Shared)
- 7.3. Delegados y Eventos Personalizados

Parte III – Windows Forms y Controles Básicos

Capítulo 8. Introducción a Windows Forms

- 8.1. Crear un Formulario Básico
- 8.2. Ciclo de Vida de un Formulario
- 8.3. Propiedades Principales de los Formularios

Capítulo 9. Controles Básicos en Windows Forms

- 9.1. Label
- 9.2. TextBox
- 9.3. Button
- 9.4. CheckBox y RadioButton
- 9.5. ListBox y ComboBox
- 9.6. PictureBox
- 9.7. GroupBox y Panel

Capítulo 10. Propiedades de Controles y Formularios

- 10.1. Propiedades comunes: Name, Text, Enabled, Visible
- 10.2. Propiedades de diseño: Font, BackColor, ForeColor
- 10.3. Propiedades de disposición: Size, Location, Dock, Anchor
- 10.4. Métodos comunes: Focus, Select, Refresh

Capítulo 11. Modelo de Eventos y Manipulación

- 11.1. Modelo de eventos en VB.NET
- 11.2. Eventos de ratón (Click, DoubleClick, MouseDown, MouseUp, MouseMove)
- 11.3. Eventos de teclado (KeyDown, KeyUp, KeyPress)
- 11.4. Eventos de carga, activación y cierre de formularios
- 11.5. Crear y manejar eventos personalizados (AddHandler, RaiseEvent)

Parte IV – Controles Avanzados

Capítulo 12. Menús y Barras de Herramientas

- 12.1. MenuStrip: creación de menús y submenús
- 12.2. ToolStrip: botones e iconos en barras de herramientas
- 12.3. ContextMenuStrip: menús contextuales
- 12.4. Menús dinámicos en tiempo de ejecución

Capítulo 13. Controles de Organización y Navegación

- 13.1. TabControl (pestañas)
- 13.2. SplitContainer (paneles redimensionables)
- 13.3. TreeView (información jerárquica)
- 13.4. ListView (datos en formato lista y detalles)

Capítulo 14. Controles de Datos

- 14.1. DataGridView
- 14.2. Enlaces de datos (Data Binding)
- 14.3. Cargar y mostrar datos en tablas

Parte V – Funcionalidades Específicas

Capítulo 15. Funciones y Procedimientos

- 15.1. Diferencia entre Function y Sub
- 15.2. Paso de parámetros por valor (ByVal) y por referencia (ByRef)
- 15.3. Funciones integradas de VB.NET (Matemáticas, Cadenas, Conversión)
- 15.4. Funciones recursivas (Ejemplo: Factorial)

Capítulo 16. Mensajes y Diálogos

- 16.1. Uso de MsgBox
- 16.2. MessageBox con botones y estilos
- 16.3. Cuadros de diálogo comunes
    - 16.3.1. OpenFileDialog
    - 16.3.2. SaveFileDialog
    - 16.3.3. ColorDialog
    - 16.3.4. FontDialog

Capítulo 17. Manejo de Texto y Formato

- 17.1. Manipulación de cadenas (String, StringBuilder)
- 17.2. Formato de cadenas y números (String.Format, interpolación)
- 17.3. Funciones de conversión de datos (CInt, CDbl, CStr, etc.)
- 17.4. Funciones de validación (String.IsNullOrWhiteSpace, Regex, TryParse)

Capítulo 18. Manejo de Fechas y Horas

- 18.1. Declaración de variables Date
- 18.2. Propiedades y métodos de DateTime (Year, Day, AddDays, etc.)
- 18.3. Operaciones con fechas (sumar, restar, comparar)
- 18.4. Formato de fecha y hora

Parte VI (Implícita) – Formularios Múltiples y Comunicación

Capítulo 19. Creación y Uso de Múltiples Formularios

- 19.1. Abrir y cerrar formularios secundarios
- 19.2. Formularios modales (ShowDialog) y no modales (Show)
- 19.3. Pasar datos entre formularios (Usando constructor o propiedades públicas)
- 19.4. Uso de propiedades y métodos entre formularios

Capítulo 20. Comunicación entre Controles de Distintos Formularios

- 20.1. Enviar valores de TextBox de un formulario a otro
- 20.2. Actualizar controles en formularios abiertos
- 20.3. Formularios MDI (Multiple Document Interface)

Parte VII – Temas Avanzados y Proyectos

Capítulo 21. Manejo de Archivos y Directorios

- 21.1. Lectura y escritura de archivos de texto (File.ReadAllText, File.WriteAllText)
- 21.2. Uso de StreamReader y StreamWriter
- 21.3. Manejo de archivos binarios (BinaryReader, BinaryWriter)

Capítulo 22. Manejo de Errores y Depuración

- 22.1. Try, Catch, Finally
    - Ejemplo: Captura de `DivideByZeroException`
- 22.2. Excepciones personalizadas
- 22.3. Herramientas de depuración en Visual Studio (Breakpoints, Step Into, Step Over, Ventanas de depuración)

Capítulo 23. Proyectos Prácticos con Windows Forms

- 23.1. Calculadora con TextBox, Label y Button
- 23.2. Agenda de contactos con interacción entre formularios
- 23.3. Aplicación de inventario con DataGridView
- 23.4. Bloc de notas con menús y ToolStrip
- 23.5. Reproductor multimedia con botones y diálogos

Parte I – Fundamentos de la Programación en VB.NET

Capítulo 1. Introducción a Visual Basic .NET

Visual Basic .NET (VB.NET) es un lenguaje de programación desarrollado por Microsoft que forma parte de la plataforma .NET. Su propósito principal es facilitar el desarrollo de aplicaciones de escritorio, aplicaciones web, servicios y, en general, software robusto y escalable. Se caracteriza por su sintaxis clara y su enfoque orientado a objetos, lo que lo hace accesible tanto a principiantes como a desarrolladores con experiencia.

VB.NET evolucionó a partir de **Visual Basic clásico (VB6 y anteriores)**, incorporando nuevas características modernas, soporte para la programación orientada a objetos completa, manejo de excepciones, uso de librerías avanzadas y compatibilidad con múltiples plataformas gracias a .NET.

1.1. Historia y evolución de Visual Basic

- **1991 – Visual Basic 1.0**: Microsoft lanza la primera versión orientada a crear aplicaciones Windows de forma visual, arrastrando y soltando controles en formularios.
- **Década de 1990 – Versiones 2.0 a 6.0**: Visual Basic se convierte en uno de los lenguajes más usados en desarrollo de aplicaciones de escritorio y bases de datos. Su sencillez y la integración con Access y SQL Server le dieron gran popularidad.
- **2002 – Nace Visual Basic .NET**: Microsoft lanza .NET Framework 1.0 y, con él, Visual Basic .NET, rompiendo compatibilidad con VB6 para ofrecer un lenguaje totalmente orientado a objetos.
- **2005–2010 – Evolución con .NET Framework**: Nuevas características como genéricos, LINQ, integración con WPF y WCF, mejoras en la depuración y desarrollo de aplicaciones web.
- **2016 – Visual Basic en .NET Core**: Microsoft abre el ecosistema .NET como multiplataforma (Windows, Linux, macOS), y aunque C# se convierte en el principal lenguaje, Visual Basic sigue disponible.
- **Actualidad – .NET 5, 6, 7, 8**: VB.NET continúa existiendo como opción, especialmente para aplicaciones Windows Forms, aunque Microsoft centra la innovación en C#. Aun así, VB.NET sigue siendo muy útil para quienes buscan un entorno sencillo y productivo.

1.2. El entorno .NET Framework y .NET Core

- **.NET Framework**: Plataforma original lanzada en 2002, diseñada principalmente para Windows. Incluye una gran colección de librerías (Framework Class Library – FCL) y el Common Language Runtime (CLR), que ejecuta el código gestionado.
- **.NET Core**: Introducido en 2016, es una versión de .NET de código abierto y multiplataforma, compatible con Windows, Linux y macOS. Está optimizado para aplicaciones modernas, alto rendimiento y despliegue en la nube.
- **.NET (5 en adelante)**: Microsoft unificó .NET Framework y .NET Core en una única plataforma llamada simplemente ".NET", disponible desde 2020 en adelante. Permite desarrollar aplicaciones de escritorio, web, móviles y servicios en un solo ecosistema.

En el caso de **Windows Forms**, el soporte sigue siendo sólido tanto en .NET Framework como en .NET 6/7/8, por lo que es posible elegir según las necesidades del proyecto.

1.3. Instalación y configuración de Visual Studio

Visual Studio es el IDE oficial para desarrollar en Visual Basic .NET. Ofrece herramientas gráficas, depuración avanzada, integración con bases de datos y diseñadores visuales para Windows Forms.

Pasos para instalarlo:

1. Descargar Visual Studio Community (gratuito para estudiantes, desarrolladores individuales y pequeñas empresas) desde la web de Microsoft.
2. Durante la instalación, seleccionar la carga de trabajo **"Desarrollo de escritorio con .NET"**.
3. Instalar también SQL Server Express si se desea trabajar con bases de datos locales.
4. Configurar el idioma y tema visual del IDE (claro, oscuro o azul).
5. Una vez instalado, se puede crear un nuevo proyecto de **Aplicación Windows Forms (.NET Framework o .NET 6/7)**.

1.4. Estructura de un proyecto de Windows Forms

Cuando se crea un proyecto de Windows Forms en VB.NET, se genera una estructura básica de archivos y carpetas:

- **My Project**: Contiene la configuración general del proyecto (nombre, ensamblado, versión, recursos).
- **Form1.vb**: Archivo principal del formulario por defecto. Contiene tanto el diseño gráfico como el código asociado.
- **Form1.Designer.vb**: Archivo generado automáticamente que define los controles (botones, cuadros de texto, etiquetas) añadidos al formulario.
- **Module1.vb** (opcional): Archivo donde se pueden incluir funciones y procedimientos generales.

- **bin/**: Carpeta donde se almacenan los archivos compilados ejecutables (.exe) y librerías (.dll).
- **obj/**: Carpeta temporal con archivos intermedios de compilación.

Ejemplo de código básico en un formulario:

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        MessageBox.Show("¡Hola, mundo en VB.NET!")
    End Sub
End Class
```

Este pequeño ejemplo muestra cómo, con un solo botón en un formulario, se puede interactuar con el usuario mediante un cuadro de diálogo.

Capítulo 2. Conceptos Básicos de Programación

La programación en Visual Basic .NET se basa en la combinación de elementos de sintaxis simples y estructuras claras que permiten construir desde programas básicos hasta aplicaciones complejas. Antes de avanzar hacia temas más avanzados como estructuras de control, clases o bases de datos, es fundamental comprender cómo se organiza un programa, cómo se definen sus bloques de instrucciones y la importancia de los comentarios para mantener un código legible.

2.1. Estructura de un programa en VB.NET

Un programa en VB.NET, dentro de un proyecto de **Windows Forms**, se compone de módulos, clases y métodos.

La **estructura mínima** de un formulario suele verse de la siguiente manera:

```
Public Class Form1

    ' Constructor: se ejecuta al inicializar el formulario
    Public Sub New()
        InitializeComponent()
    End Sub

    ' Ejemplo de evento: cuando se hace clic en un botón
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        MessageBox.Show("Bienvenido al mundo de VB.NET")
    End Sub

End Class
```

Elementos principales:

1. **Declaración de clase (`Public Class ... End Class`)**
   Todo formulario es en realidad una clase que hereda de `System.Windows.Forms.Form`. Dentro de ella se definen los controles y la lógica del programa.

2. **Constructor (`Sub New()`)**
   Se ejecuta cuando se crea la instancia del formulario. Dentro de él siempre se llama a `InitializeComponent()`, que carga y dibuja todos los controles definidos en el diseñador visual.

3. **Procedimientos y funciones**
   Se pueden definir procedimientos (`Sub`) y funciones (`Function`) que agrupan instrucciones.

4. **Eventos**
   Cada control (botón, cuadro de texto, menú) genera eventos que el programador puede manejar escribiendo código en sus métodos asociados. Ejemplo: `Button1_Click` se ejecuta al hacer clic en el botón.

En proyectos más grandes, se pueden tener múltiples formularios, módulos con funciones globales y clases personalizadas.

2.2. Sentencias, bloques e indentación

En VB.NET, **cada instrucción o sentencia se escribe en una línea** y finaliza de manera implícita al terminar esa línea.

Ejemplo de sentencias simples:

```
Dim nombre As String = "Ana"
Dim edad As Integer = 25
MessageBox.Show("Nombre: " & nombre & " | Edad: " & edad)
```

Bloques de código

Un bloque agrupa varias sentencias relacionadas y se delimita con palabras clave de apertura y cierre. Ejemplos:

- **Bloque If...End If:**

```
If edad >= 18 Then
    MessageBox.Show("Eres mayor de edad")
Else
    MessageBox.Show("Eres menor de edad")
End If
```

- **Bloque For...Next:**

```
For i As Integer = 1 To 5
    MessageBox.Show("Iteración número: " & i)
Next
```

- **Bloque Select Case:**

```
Select Case edad
    Case Is < 18
        MessageBox.Show("Menor de edad")
    Case 18 To 65
        MessageBox.Show("Adulto")
    Case Else
        MessageBox.Show("Adulto mayor")
End Select
```

Indentación

La **indentación** consiste en sangrar las líneas hacia la derecha para indicar visualmente qué instrucciones pertenecen a qué bloque. Aunque Visual Basic .NET no obliga a indentar, es **buena práctica** hacerlo porque facilita la lectura y el mantenimiento.

Ejemplo con indentación correcta:

```
If edad >= 18 Then
    MessageBox.Show("Mayor de edad")
Else
    MessageBox.Show("Menor de edad")
End If
```

Ejemplo sin indentación (confuso):

```
If edad >= 18 Then
MessageBox.Show("Mayor de edad")
Else
MessageBox.Show("Menor de edad")
End If
```

2.3. Uso de comentarios en el código

Los **comentarios** son textos incluidos en el código que no se ejecutan, pero sirven para explicar lo que hace un fragmento del programa.

En VB.NET se escriben con un **apóstrofo (`'`)** al inicio de la línea o después de una instrucción.

Ejemplos:

```
' Este es un comentario de línea completa
Dim nombre As String = "Carlos" ' Comentario al final de la instrucción
```

Buenas prácticas en el uso de comentarios

- Explicar la **intención del código**, no repetir lo que ya se entiende.
- Colocar comentarios en bloques largos para guiar al lector.
- Evitar comentarios innecesarios que hagan más difícil mantener el código.

Ejemplo práctico con comentarios útiles:

```
' Programa sencillo que pide el nombre y lo muestra en pantalla
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        ' Declarar una variable para almacenar el nombre
        Dim nombre As String = TextBox1.Text

        ' Verificar si el usuario escribió algo
        If nombre = "" Then
            MessageBox.Show("Por favor, introduce tu nombre")
        Else
            ' Mostrar un saludo personalizado
            MessageBox.Show("Hola, " & nombre & " ¡Bienvenido!")
        End If
    End Sub

End Class
```

Aquí los comentarios aclaran cada paso sin repetir lo obvio, ayudando a comprender el flujo del programa.

Capítulo 3. Variables y Tipos de Datos

En cualquier lenguaje de programación, las **variables** son la base para almacenar y manipular información. Visual Basic .NET permite manejar diferentes tipos de datos de manera flexible y clara, siendo esencial comprender cómo declararlas, asignarles valores y controlar su tipo.

3.1. Declaración y asignación de variables

En VB.NET, una variable se declara utilizando la palabra clave `Dim` seguida del nombre y opcionalmente del tipo de dato. Posteriormente, se le puede asignar un valor.

Ejemplo básico:

```
Dim nombre As String
Dim edad As Integer

nombre = "Ana"
edad = 25

MessageBox.Show("Nombre: " & nombre & " | Edad: " & edad)
```

También se puede **declarar y asignar en la misma línea**:

```
Dim ciudad As String = "Zaragoza"
Dim temperatura As Double = 23.5
```

Reglas para nombres de variables:

- No pueden empezar con un número.
- No pueden contener espacios ni caracteres especiales (excepto _).
- No deben coincidir con palabras reservadas de VB.NET (`If`, `End`, `Sub`, etc.).

3.2. Tipos de datos primitivos y compuestos

Tipos primitivos más comunes:

Tipo	Descripción	Ejemplo
`Integer`	Números enteros	10, -5
`Double`	Números con decimales	3.14, -0.5
`String`	Cadenas de texto	"Hola Mundo"
`Boolean`	Verdadero o falso	True, False
`Char`	Un solo carácter	"A"

Tipos compuestos:

- `Array`: Arreglos de datos del mismo tipo.

```
Dim numeros(4) As Integer ' Arreglo de 5 elementos
numeros(0) = 10
numeros(1) = 20
```

- `List(Of T)`: Lista dinámica de elementos.

```
Dim nombres As New List(Of String)
nombres.Add("Ana")
nombres.Add("Carlos")
```

- `Structure` y `Class`: Para agrupar datos y métodos relacionados.

3.3. Variables estáticas y dinámicas

Variables estáticas

- Mantienen su valor entre distintas llamadas a un procedimiento.
- Se declaran con `Static` dentro de un `Sub` o `Function`.

```
Private Sub ContadorEst()
    Static contador As Integer = 0
    contador += 1
    MessageBox.Show("Contador: " & contador)
End Sub
```

Cada vez que se ejecuta `ContadorEst`, el valor anterior de `contador` se conserva.

Variables dinámicas

- Se crean y eliminan automáticamente según el alcance del bloque.
- Declaradas normalmente con `Dim`.

3.4. Conversión y validación de tipos

VB.NET permite **convertir un tipo de dato a otro** mediante funciones integradas:

- `CInt(valor)`: Convierte a entero.
- `CDbl(valor)`: Convierte a doble.
- `CStr(valor)`: Convierte a cadena.
- `CBool(valor)`: Convierte a booleano.
-

Ejemplo práctico:

```
Dim texto As String = "123"
Dim numero As Integer

' Validación antes de la conversión
If IsNumeric(texto) Then
    numero = CInt(texto)
    MessageBox.Show("Número convertido: " & numero)
Else
    MessageBox.Show("No es un número válido")
End If
```

3.5. Constantes y enumeraciones

Constantes

Son valores **fijos** que no cambian durante la ejecución del programa. Se declaran con `Const`.

```
Const Pi As Double = 3.1416
Const MensajeBienvenida As String = "Bienvenido al programa"
```

Enumeraciones

Permiten **nombrar un conjunto de valores enteros**, mejorando la legibilidad del código.

```
Enum DiasSemana
    Lunes = 1
    Martes
    Miercoles
    Jueves
    Viernes
    Sabado
    Domingo
End Enum

Dim hoy As DiasSemana
hoy = DiasSemana.Miercoles
MessageBox.Show("Hoy es el día número: " & hoy)
```

Ejemplo práctico en un Formulario:

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        ' Declaración de variables
        Dim nombre As String = TextBox1.Text
        Dim edad As Integer

        ' Validación y conversión
        If IsNumeric(TextBox2.Text) Then
            edad = CInt(TextBox2.Text)
        Else
            MessageBox.Show("Introduce un número válido")
            Exit Sub
        End If

        ' Constante
        Const mensaje As String = "Datos del usuario:"

        ' Mostrar información
        MessageBox.Show(mensaje & vbCrLf & "Nombre: " & nombre & vbCrLf & "Edad:
" & edad)
    End Sub

End Class
```

Este formulario pide al usuario **nombre y edad**, valida que la edad sea numérica y muestra los datos usando variables, conversión y una constante.

Capítulo 4. Operadores y Expresiones

Los operadores y expresiones son esenciales para realizar cálculos, comparaciones y tomar decisiones en un programa. VB.NET ofrece un amplio conjunto de operadores para trabajar con distintos tipos de datos y estructuras.

4.1. Operadores aritméticos, lógicos y relacionales

Operadores aritméticos

Se usan para realizar cálculos matemáticos.

Operador	Descripción	Ejemplo
`+`	Suma	5 + 3 = 8
`-`	Resta	5 - 3 = 2
`*`	Multiplicación	5 * 3 = 15
`/`	División	10 / 2 = 5
`\`	División entera	10 \ 3 = 3
`Mod`	Módulo (resto)	10 Mod 3 = 1
`^`	Potencia	2 ^ 3 = 8

Operadores relacionales

Se usan para comparar valores y devuelven un valor Booleano (`True` o `False`).

Operador	Descripción	Ejemplo
`=`	Igual que	5 = 5 → True
`<>`	Diferente de	5 <> 3 → True
`>`	Mayor que	5 > 3 → True
`<`	Menor que	5 < 3 → False
`>=`	Mayor o igual que	5 >= 5 → True
`<=`	Menor o igual que	3 <= 5 → True

Operadores lógicos

Se usan para combinar expresiones Booleanas.

Operador	Descripción	Ejemplo
`And`	Y lógico	True And False → False
`Or`	O lógico	True Or False → True
`Not`	Negación	Not True → False
`AndAlso`	Y lógico de corto circuito	Evalúa solo si necesario
`OrElse`	O lógico de corto circuito	Evalúa solo si necesario

4.2. Operadores de asignación

Se usan para **asignar valores a variables** y se combinan con operadores aritméticos.

Operador	Descripción	Ejemplo
`=`	Asignación simple	x = 5
`+=`	Suma y asignación	x += 3 → x = x + 3
`-=`	Resta y asignación	x -= 2 → x = x - 2
`*=`	Multiplicación y asignación	x *= 4 → x = x * 4
`/=`	División y asignación	x /= 2 → x = x / 2

4.3. Operadores de concatenación de cadenas

En VB.NET, se usan para unir cadenas de texto:

Operador	Descripción	Ejemplo
`&`	Concatenación de cadenas	"Hola " & "Mundo" → "Hola Mundo"
`+`	También concatena cadenas	"Hola " + "VB.NET" → "Hola VB.NET"

Ejemplo práctico:

```
Dim nombre As String = "Ana"
Dim saludo As String = "Hola " & nombre & ", bienvenido!"
MessageBox.Show(saludo)
```

4.4. Expresiones condicionales

VB.NET permite **evaluar expresiones y tomar decisiones** mediante estructuras condicionales (`If`, `ElseIf`, `Else`) y el operador ternario `If`.

Estructura `If` básica:

```
Dim edad As Integer = 20


If edad >= 18 Then
    MessageBox.Show("Eres mayor de edad")
Else
    MessageBox.Show("Eres menor de edad")
End If
```

Operador ternario `If` (inline):

```
Dim edad As Integer = 16
Dim mensaje As String = If(edad >= 18, "Mayor de edad", "Menor de edad")
MessageBox.Show(mensaje)
```

Ejemplo práctico en Windows Forms

```
Public Class Form1


    Private Sub btnCalcular_Click(sender As Object, e As EventArgs) Handles
btnCalcular.Click

        Dim numero1 As Double

        Dim numero2 As Double

        Dim suma As Double


        ' Validación de entrada

        If Not Double.TryParse(txtNumero1.Text, numero1) Or Not
Double.TryParse(txtNumero2.Text, numero2) Then

            MessageBox.Show("Introduce números válidos")

            Exit Sub

        End If


        ' Operadores aritméticos y de asignación

        suma = numero1 + numero2


        ' Operadores relacionales y lógicos

        If suma > 100 And suma < 500 Then

            MessageBox.Show("La suma está entre 100 y 500: " & suma)

        Else

            MessageBox.Show("La suma no está en el rango: " & suma)

        End If


        ' Concatenación de cadenas

        Dim resultadoTexto As String = "Resultado de " & numero1 & " + " &
numero2 & " = " & suma

        MessageBox.Show(resultadoTexto)

    End Sub


End Class
```

En este formulario, el usuario introduce dos números, se validan, se suman y se muestran mensajes usando **operadores aritméticos, lógicos, relacionales y de concatenación**.

Capítulo 5. Estructuras de Control

Las estructuras de control permiten dirigir el flujo de ejecución de un programa según condiciones o repeticiones. Son esenciales para la lógica de cualquier aplicación.

5.1. Condicionales: If, ElseIf, Select Case

`If … Then … Else`

Se usa para ejecutar código según una condición:

```
Dim edad As Integer = 20

If edad >= 18 Then
    MessageBox.Show("Eres mayor de edad")
Else
    MessageBox.Show("Eres menor de edad")
End If
```

`ElseIf`

Permite evaluar múltiples condiciones:

```
Dim nota As Integer = 75

If nota >= 90 Then
    MessageBox.Show("Excelente")
ElseIf nota >= 70 Then
    MessageBox.Show("Bueno")
ElseIf nota >= 50 Then
    MessageBox.Show("Suficiente")
Else
    MessageBox.Show("Insuficiente")
End If
```

`Select Case`

Útil cuando se deben evaluar múltiples valores de una variable:

```
Dim dia As Integer = 3

Select Case dia
    Case 1
        MessageBox.Show("Lunes")
    Case 2
        MessageBox.Show("Martes")
    Case 3
        MessageBox.Show("Miércoles")
    Case Else
        MessageBox.Show("Otro día")
End Select
```

5.2. Bucles: For, While, Do Until

Bucle `For`

Se utiliza cuando se conoce el número de iteraciones:

```
For i As Integer = 1 To 5
    MessageBox.Show("Número: " & i)
Next
```

Bucle `While`

Se ejecuta mientras la condición sea verdadera:

```
Dim contador As Integer = 1

While contador <= 5
    MessageBox.Show("Contador: " & contador)
    contador += 1
End While
```

Bucle `Do Until`/`Do While`

Se ejecuta hasta que se cumpla (o deje de cumplirse) la condición:

```
Dim numero As Integer = 1

Do Until numero > 5
    MessageBox.Show("Número: " & numero)
    numero += 1
Loop
```

5.3. Estructuras anidadas

Se pueden combinar condicionales y bucles dentro de otras estructuras:

```
For i As Integer = 1 To 3
    For j As Integer = 1 To 3
        MessageBox.Show("i = " & i & ", j = " & j)
    Next
Next
```

Condicional anidada dentro de un bucle:

```
For i As Integer = 1 To 5
    If i Mod 2 = 0 Then
        MessageBox.Show(i & " es par")
    Else
        MessageBox.Show(i & " es impar")
    End If
Next
```

5.4. Uso de Exit, Continue y Return

`Exit`

Permite salir de un bucle o procedimiento antes de tiempo:

```
For i As Integer = 1 To 10
    If i = 5 Then
        Exit For ' Sale del bucle cuando i es 5
    End If
    MessageBox.Show(i)
Next
```

Continue

Permite saltar a la siguiente iteración del bucle:

```
For i As Integer = 1 To 5
    If i = 3 Then
        Continue For ' Salta la iteración cuando i es 3
    End If
    MessageBox.Show(i)
Next
```

Return

Se usa para salir de un procedimiento o función y opcionalmente devolver un valor:

```
Function Sumar(a As Integer, b As Integer) As Integer
    If a < 0 Or b < 0 Then
        Return 0 ' Sale inmediatamente
    End If
    Return a + b
End Function
```

Ejemplo práctico en Windows Forms

```
Public Class Form1

    Private Sub btnEjecutar_Click(sender As Object, e As EventArgs) Handles
btnEjecutar.Click

        ' Condicional If

        Dim edad As Integer

        If Not Integer.TryParse(txtEdad.Text, edad) Then

            MessageBox.Show("Introduce un número válido")

            Exit Sub

        End If


        If edad >= 18 Then

            MessageBox.Show("Eres mayor de edad")

        Else

            MessageBox.Show("Eres menor de edad")

        End If


        ' Bucle For

        For i As Integer = 1 To edad

            If i Mod 5 = 0 Then Continue For ' Salta múltiplos de 5

            MessageBox.Show("Número: " & i)

        Next


        ' Select Case

        Select Case edad

            Case 0 To 12

                MessageBox.Show("Niño")

            Case 13 To 19

                MessageBox.Show("Adolescente")

            Case Else

                MessageBox.Show("Adulto")

        End Select

    End Sub

End Class
```

En este formulario, se combinan **condicionales, bucles, estructuras anidadas y uso de Exit/Continue** de manera práctica para mostrar mensajes según la edad ingresada.

Ejercicios

---

1. **Programa 1 – Formulario básico con un botón**
   Enunciado: Crea un formulario con un botón que al pulsarlo muestre un mensaje "Hola Visual Basic .NET".

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        MessageBox.Show("Hola Visual Basic .NET")
    End Sub
End Class
```

2. **Programa 2 – Etiqueta con texto inicial**
   Enunciado: Muestra un formulario con una etiqueta que contiene el texto "Bienvenido a VB.NET".

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Label1.Text = "Bienvenido a VB.NET"
    End Sub
End Class
```

3. **Programa 3 – Cambiar el color de fondo**
   Enunciado: Cambia el color de fondo del formulario al hacer clic en un botón.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Me.BackColor = Color.LightBlue
    End Sub
End Class
```

4. **Programa 4 – Caja de texto y botón**
   Enunciado: Escribe un nombre en una caja de texto y al pulsar un botón se muestra en una etiqueta.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Label1.Text = "Hola " & TextBox1.Text
    End Sub
End Class
```

5. **Programa 5 – Contador con botón**
   Enunciado: Cada vez que se pulse un botón, se incrementa un contador mostrado en una etiqueta.

```
Public Class Form1
    Dim contador As Integer = 0
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        contador += 1
        Label1.Text = contador.ToString()
    End Sub
End Class
```

6. **Programa 6 – Cambiar título del formulario**
   Enunciado: Un botón que cambia el título de la ventana del formulario.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Me.Text = "Nuevo título de la ventana"
    End Sub
End Class
```

7. **Programa 7 – Salir de la aplicación**
   Enunciado: Un botón que cierra el programa al pulsarlo.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Application.Exit()
    End Sub
End Class
```

8. **Programa 8 – Cambiar tamaño de formulario**
   Enunciado: Un botón que cambia el tamaño de la ventana al pulsarlo.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Me.Size = New Size(800, 600)
    End Sub
End Class
```

9. **Programa 9 – Mostrar fecha actual**
   Enunciado: Un botón que muestra la fecha actual en una etiqueta.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Label1.Text = DateTime.Now.ToShortDateString()
    End Sub
End Class
```

10. **Programa 10 – Mostrar hora actual**
    Enunciado: Un botón que muestra la hora actual en una etiqueta.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Label1.Text = DateTime.Now.ToLongTimeString()
    End Sub
End Class
```

11. Programa 11 – Suma de dos números

Enunciado: Dos cajas de texto para introducir números y un botón que muestra la suma en una etiqueta.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim n1 As Integer = CInt(TextBox1.Text)
        Dim n2 As Integer = CInt(TextBox2.Text)
        Label1.Text = "Resultado: " & (n1 + n2).ToString()
    End Sub
End Class
```

12. Programa 12 – Cambiar color de etiqueta

Enunciado: Cambia el color del texto de una etiqueta al pulsar un botón.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Label1.ForeColor = Color.Red
    End Sub
End Class
```

13. Programa 13 – Minimizar ventana

Enunciado: Un botón que minimiza la ventana del formulario.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Me.WindowState = FormWindowState.Minimized
    End Sub
End Class
```

14. Programa 14 – Maximizar ventana

Enunciado: Un botón que maximiza la ventana del formulario.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Me.WindowState = FormWindowState.Maximized
    End Sub
End Class
```

15. Programa 15 – Reproducir sonido de sistema

Enunciado: Un botón que hace sonar un beep del sistema.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Beep()
    End Sub
End Class
```

16. **Programa 16 – Barra de progreso**

Enunciado: Un botón que avanza una barra de progreso en intervalos de 10.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        If ProgressBar1.Value < ProgressBar1.Maximum Then
            ProgressBar1.Value += 10
        End If
    End Sub
End Class
```

17. **Programa 17 – Mostrar cuadro de diálogo**

Enunciado: Un botón que abre un cuadro de mensaje con "Este es un cuadro de diálogo".

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        MessageBox.Show("Este es un cuadro de diálogo")
    End Sub
End Class
```

18. **Programa 18 – Cambiar fuente de etiqueta**

Enunciado: Cambia el tamaño y estilo de la fuente de una etiqueta.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Label1.Font = New Font("Arial", 16, FontStyle.Bold)
    End Sub
End Class
```

19. **Programa 19 – Timer en acción**

Enunciado: Un temporizador que actualiza la hora actual en una etiqueta cada segundo.

```
Public Class Form1
    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles
Timer1.Tick
        Label1.Text = DateTime.Now.ToLongTimeString()
    End Sub
End Class
```

20. **Programa 20 – ListBox con elementos**

Enunciado: Un botón que añade elementos a un ListBox.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        ListBox1.Items.Add("Elemento nuevo")
    End Sub
End Class
```

Ejercicios

---

1. Programa 1 – Formulario con mensaje simple

Enunciado: Un formulario que muestra un mensaje al cargar, mostrando la estructura básica de un programa VB.NET.

```
Public Class Form1
    ' Evento que se ejecuta al cargar el formulario
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        MessageBox.Show("Bienvenido a VB.NET")
    End Sub
End Class
```

2. Programa 2 – Botón con mensaje

Enunciado: Botón que al hacer clic muestra un mensaje, ilustrando sentencias y bloques.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        ' Mostrar mensaje al usuario
        MessageBox.Show("Has pulsado el botón")
    End Sub
End Class
```

3. Programa 3 – Uso de variables

Enunciado: Declarar variables, asignarles valores y mostrarlos en una etiqueta.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim nombre As String = "Ana"
        Dim edad As Integer = 25
        Label1.Text = "Nombre: " & nombre & ", Edad: " & edad
    End Sub
End Class
```

4. Programa 4 – Suma de dos números

Enunciado: Mostrar cómo usar bloques de código para realizar una operación y mostrar el resultado.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim a As Integer = 10
        Dim b As Integer = 20
        Dim suma As Integer = a + b
        MessageBox.Show("La suma es: " & suma)
    End Sub
End Class
```

5. Programa 5 – Bloque condicional If

Enunciado: Usar un bloque If para evaluar un valor y mostrar un mensaje.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim numero As Integer = 5
        If numero > 0 Then
            MessageBox.Show("El número es positivo")
        Else
            MessageBox.Show("El número no es positivo")
        End If
    End Sub
End Class
```

6. Programa 6 – Bloque If anidado

Enunciado: Usar bloques If anidados para evaluar varias condiciones.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim numero As Integer = 0
        If numero > 0 Then
            MessageBox.Show("Positivo")
        ElseIf numero < 0 Then
            MessageBox.Show("Negativo")
        Else
            MessageBox.Show("Cero")
        End If
    End Sub
End Class
```

7. Programa 7 – Uso de comentarios

Enunciado: Mostrar cómo documentar el código con comentarios en distintas líneas.

```
Public Class Form1
    ' Evento principal del formulario
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        ' Mostrar mensaje de bienvenida
        MessageBox.Show("Este programa usa comentarios para explicar el código")
    End Sub
End Class
```

8. Programa 8 – Bucle For

Enunciado: Usar un bucle For para mostrar los números del 1 al 5 en un ListBox.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        ListBox1.Items.Clear()
        For i As Integer = 1 To 5
            ListBox1.Items.Add("Número: " & i)
        Next
    End Sub
End Class
```

9. Programa 9 – Bucle While

Enunciado: Mostrar un bucle While que suma números hasta que el total sea mayor a 10.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim total As Integer = 0
        Dim i As Integer = 1
        While total <= 10
            total += i
            i += 1
        End While
        MessageBox.Show("Total final: " & total)
    End Sub
End Class
```

10. Programa 10 – Estructura Select Case

Enunciado: Uso de Select Case para evaluar valores de una variable.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim dia As Integer = 3
        Select Case dia
            Case 1
                MessageBox.Show("Lunes")
            Case 2
                MessageBox.Show("Martes")
            Case 3
                MessageBox.Show("Miércoles")
            Case Else
                MessageBox.Show("Otro día")
        End Select
    End Sub
End Class
```

11. Programa 11 – Función simple

Enunciado: Crear una función que devuelve el doble de un número.

```
Public Class Form1
    Private Function Doble(numero As Integer) As Integer
        Return numero * 2
    End Function

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        MessageBox.Show("Doble de 5: " & Doble(5))
    End Sub
End Class
```

12. Programa 12 – Procedimiento Sub

Enunciado: Usar un procedimiento Sub para mostrar un mensaje.

```
Public Class Form1
    Private Sub MostrarMensaje()
        MessageBox.Show("Mensaje desde un procedimiento Sub")
    End Sub
```

```
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        MostrarMensaje()
    End Sub
End Class
```

13. Programa 13 – Operaciones con variables

Enunciado: Declarar varias variables, realizar operaciones y mostrar el resultado.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim x As Integer = 10
        Dim y As Integer = 3
        Dim resultado As Integer
        resultado = x * y + 2
        MessageBox.Show("Resultado: " & resultado)
    End Sub
End Class
```

14. Programa 14 – Concatenación de texto

Enunciado: Combinar varias cadenas usando el operador & y mostrar en un Label.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim nombre As String = "Ana"
        Dim apellido As String = "Pérez"
        Label1.Text = "Nombre completo: " & nombre & " " & apellido
    End Sub
End Class
```

15. Programa 15 – Declaración de constantes

Enunciado: Usar constantes para definir valores fijos en el programa.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Const PI As Double = 3.1416
        MessageBox.Show("Valor de PI: " & PI)
    End Sub
End Class
```

16. Programa 16 – Uso de comentarios multilínea

Enunciado: Explicar varias líneas de código usando comentarios.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        ' Declaramos variables
        Dim a As Integer = 5
        Dim b As Integer = 10
        ' Calculamos la suma
        Dim suma As Integer = a + b
        ' Mostramos el resultado
```

```
        MessageBox.Show("Suma: " & suma)
    End Sub
End Class
```

17. Programa 17 – Indentación correcta

Enunciado: Ejemplo de buena indentación usando If y bucles.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        For i As Integer = 1 To 3
            If i Mod 2 = 0 Then
                MessageBox.Show(i & " es par")
            Else
                MessageBox.Show(i & " es impar")
            End If
        Next
    End Sub
End Class
```

18. Programa 18 – Comentarios explicativos en formularios

Enunciado: Documentar el formulario y cada control con comentarios.

```
Public Class Form1
    ' Este formulario sirve para mostrar ejemplos de programación
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        ' Mostrar mensaje de ejemplo
        MessageBox.Show("Ejemplo de comentario en botón")
    End Sub
End Class
```

Ejercicios

PROGRAMA 1: Declaración y asignación de variables simples

Enunciado: Muestra cómo declarar variables de tipo entero y cadena, asignarles valores y mostrarlos en un `Label`.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim numero As Integer
        Dim texto As String

        numero = 10
        texto = "Hola VB.NET"

        Label1.Text = "Número: " & numero & ", Texto: " & texto
    End Sub
End Class
```

PROGRAMA 2: Declaración múltiple de variables

Enunciado: Declara varias variables en una línea y muestra su valor.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim a, b, c As Integer
        a = 5
        b = 10
        c = 15
        Label1.Text = "a=" & a & ", b=" & b & ", c=" & c
    End Sub
End Class
```

PROGRAMA 3: Tipos de datos primitivos

Enunciado: Muestra distintos tipos de datos primitivos: entero, decimal, booleano, caracter.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim entero As Integer = 25
        Dim decimal As Double = 3.14
        Dim booleano As Boolean = True
        Dim caracter As Char = "A"c

        Label1.Text = "Entero=" & entero & ", Decimal=" & decimal & ",
Booleano=" & booleano & ", Caracter=" & caracter
    End Sub
End Class
```

PROGRAMA 4: Tipos de datos compuestos

Enunciado: Uso de arreglos y cadenas.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim numeros() As Integer = {1, 2, 3, 4, 5}
        Dim nombres() As String = {"Ana", "Luis", "Carlos"}

        Label1.Text = "Primer número: " & numeros(0) & ", Primer nombre: " &
nombres(0)
    End Sub
End Class
```

PROGRAMA 5: Variables estáticas

Enunciado: Muestra cómo una variable conserva su valor entre llamadas.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Static contador As Integer
        contador += 1
        Label1.Text = "Contador: " & contador
    End Sub
End Class
```

PROGRAMA 6: Variables dinámicas

Enunciado: Uso de `Object` para almacenar distintos tipos de datos.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim dinamico As Object
        dinamico = 10
        Label1.Text = "Valor inicial: " & dinamico
        dinamico = "VB.NET"
        Label2.Text = "Nuevo valor: " & dinamico
    End Sub
End Class
```

PROGRAMA 7: Conversión de tipos

Enunciado: Convierte un texto a número y realiza una operación.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim texto As String = "20"
        Dim numero As Integer
        numero = CInt(texto) + 10
        Label1.Text = "Resultado: " & numero
    End Sub
End Class
```

PROGRAMA 8: Validación de tipos

Enunciado: Comprueba si un valor se puede convertir a entero.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim valor As String = TextBox1.Text
        Dim numero As Integer

        If Integer.TryParse(valor, numero) Then
            Label1.Text = "Número válido: " & numero
        Else
            Label1.Text = "No es un número"
        End If
    End Sub
End Class
```

PROGRAMA 9: Constantes

Enunciado: Declara constantes y las muestra en pantalla.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Const PI As Double = 3.1416
        Const SALUDO As String = "Hola Mundo"

        Label1.Text = SALUDO & ", PI=" & PI
    End Sub
End Class
```

PROGRAMA 10: Enumeraciones

Enunciado: Definición y uso de enumeraciones.

```
Public Class Form1
    Enum DiasSemana
        Lunes = 1
        Martes
        Miercoles
        Jueves
        Viernes
    End Enum

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim hoy As DiasSemana
        hoy = DiasSemana.Miercoles
        Label1.Text = "Hoy es: " & hoy.ToString()
    End Sub
End Class
```

PROGRAMA 11: Suma de variables

Enunciado: Suma dos variables enteras y muestra el resultado.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim a As Integer = 5
        Dim b As Integer = 10
        Dim suma As Integer = a + b
        Label1.Text = "Suma: " & suma
    End Sub
End Class
```

PROGRAMA 12: Concatenación de cadenas

Enunciado: Une varias cadenas y las muestra en un `Label`.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim nombre As String = "Ana"
        Dim apellido As String = "Pérez"
        Dim completo As String = nombre & " " & apellido
        Label1.Text = "Nombre completo: " & completo
    End Sub
End Class
```

PROGRAMA 13: Conversión a cadena

Enunciado: Convierte un número a cadena para mostrarlo junto a texto.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim numero As Integer = 100
        Label1.Text = "El número es: " & numero.ToString()
    End Sub
End Class
```

PROGRAMA 14: Uso de Booleanos

Enunciado: Muestra el valor de una variable booleana en un `Label`.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim esVerdad As Boolean = True
        Label1.Text = "Valor booleano: " & esVerdad
    End Sub
End Class
```

PROGRAMA 15: Uso de caracteres

Enunciado: Muestra la manipulación de caracteres individuales.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim letra As Char = "B"c
        Label1.Text = "La letra es: " & letra
    End Sub
End Class
```

PROGRAMA 16: Array dinámico

Enunciado: Agrega elementos a un `List(Of Integer)` y los muestra.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim lista As New List(Of Integer)
        lista.Add(10)
        lista.Add(20)
        lista.Add(30)
        Label1.Text = "Primer elemento: " & lista(0) & ", Total: " & lista.Count
    End Sub
End Class
```

PROGRAMA 17: Constante con cálculo

Enunciado: Calcula el área de un círculo usando una constante.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Const PI As Double = 3.1416
        Dim radio As Double = 5
        Dim area As Double = PI * radio * radio
        Label1.Text = "Área del círculo: " & area
    End Sub
End Class
```

PROGRAMA 18: Enum con selección

Enunciado: Selecciona un valor de una enumeración y muestra un mensaje.

```
Public Class Form1
    Enum Colores
        Rojo
        Verde
        Azul
    End Enum

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim colorFavorito As Colores = Colores.Azul
        Label1.Text = "Color favorito: " & colorFavorito.ToString()
    End Sub
End Class
```

PROGRAMA 19: Variables locales y globales

Enunciado: Diferencia entre variables locales y variables de clase.

```
Public Class Form1
    Dim globalVar As Integer = 50

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim localVar As Integer = 10
        Label1.Text = "Local: " & localVar & ", Global: " & globalVar
    End Sub
End Class
```

PROGRAMA 20: Conversión segura y cálculo

Enunciado: Convierte un texto a número y evita errores si la conversión falla.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim texto As String = TextBox1.Text
        Dim numero As Integer

        If Integer.TryParse(texto, numero) Then
            Label1.Text = "Número x2: " & (numero * 2)
        Else
            Label1.Text = "Entrada inválida"
        End If
    End Sub
End Class
```

Ejercicios

Programa 1: Suma de dos números

Enunciado: Este programa solicita al usuario dos números y muestra su suma utilizando operadores aritméticos.

```
Public Class Form1
    Private Sub btnSumar_Click(sender As Object, e As EventArgs) Handles
btnSumar.Click
        Dim num1 As Integer = CInt(txtNum1.Text)
        Dim num2 As Integer = CInt(txtNum2.Text)
        Dim suma As Integer = num1 + num2
        MessageBox.Show("La suma es: " & suma)
    End Sub
End Class
```

Programa 2: Comparación de números

Enunciado: Compara dos números y muestra cuál es mayor usando operadores relacionales.

```
Public Class Form1
    Private Sub btnComparar_Click(sender As Object, e As EventArgs) Handles
btnComparar.Click
        Dim a As Integer = CInt(txtA.Text)
        Dim b As Integer = CInt(txtB.Text)
        If a > b Then
            MessageBox.Show("El mayor es: " & a)
        ElseIf b > a Then
            MessageBox.Show("El mayor es: " & b)
        Else
            MessageBox.Show("Ambos son iguales")
        End If
    End Sub
End Class
```

Programa 3: Número par o impar

Enunciado: Determina si un número ingresado es par o impar usando operadores lógicos y condicionales.

```
Public Class Form1
    Private Sub btnParImpar_Click(sender As Object, e As EventArgs) Handles
btnParImpar.Click
        Dim num As Integer = CInt(txtNumero.Text)
        If num Mod 2 = 0 Then
            MessageBox.Show("El número es par")
        Else
            MessageBox.Show("El número es impar")
        End If
    End Sub
End Class
```

Programa 4: Multiplicación y asignación

Enunciado: Multiplica un número por 5 usando un operador de asignación compuesto.

```
Public Class Form1
    Private Sub btnMultiplicar_Click(sender As Object, e As EventArgs) Handles
btnMultiplicar.Click
        Dim num As Integer = CInt(txtNum.Text)
        num *= 5
        MessageBox.Show("Resultado: " & num)
    End Sub
End Class
```

Programa 5: Concatenación de nombres

Enunciado: Une el nombre y apellido del usuario usando el operador &.

```
Public Class Form1
    Private Sub btnConcatenar_Click(sender As Object, e As EventArgs) Handles
btnConcatenar.Click
        Dim nombre As String = txtNombre.Text
        Dim apellido As String = txtApellido.Text
        Dim completo As String = nombre & " " & apellido
        MessageBox.Show("Nombre completo: " & completo)
    End Sub
End Class
```

Programa 6: Verificación de edad

Enunciado: Verifica si la persona es mayor de edad usando operadores relacionales.

```
Public Class Form1
    Private Sub btnEdad_Click(sender As Object, e As EventArgs) Handles
btnEdad.Click
        Dim edad As Integer = CInt(txtEdad.Text)
        If edad >= 18 Then
            MessageBox.Show("Eres mayor de edad")
        Else
            MessageBox.Show("Eres menor de edad")
        End If
    End Sub
End Class
```

Programa 7: Operaciones aritméticas completas

Enunciado: Realiza suma, resta, multiplicación y división de dos números.

```
Public Class Form1
    Private Sub btnOperar_Click(sender As Object, e As EventArgs) Handles
btnOperar.Click
        Dim x As Double = CDbl(txtX.Text)
        Dim y As Double = CDbl(txtY.Text)
        Dim suma As Double = x + y
        Dim resta As Double = x - y
        Dim mult As Double = x * y
        Dim div As Double
        If y <> 0 Then
            div = x / y
        Else
            div = 0
        End If
        MessageBox.Show("Suma: " & suma & vbCrLf & "Resta: " & resta & vbCrLf &
"Multiplicación: " & mult & vbCrLf & "División: " & div)
    End Sub
End Class
```

Programa 8: Comparación múltiple con `AndAlso`

Enunciado: Verifica si un número está entre 10 y 20 usando operadores lógicos.

```
Public Class Form1
    Private Sub btnVerificar_Click(sender As Object, e As EventArgs) Handles
btnVerificar.Click
        Dim num As Integer = CInt(txtNumero.Text)
        If num >= 10 AndAlso num <= 20 Then
            MessageBox.Show("El número está entre 10 y 20")
        Else
            MessageBox.Show("El número está fuera del rango")
        End If
    End Sub
End Class
```

Programa 9: Uso de operador `OrElse`

Enunciado: Verifica si un número es negativo o mayor que 100.

```
Public Class Form1
    Private Sub btnVerificar2_Click(sender As Object, e As EventArgs) Handles
btnVerificar2.Click
        Dim num As Integer = CInt(txtNumero.Text)
        If num < 0 OrElse num > 100 Then
            MessageBox.Show("Número fuera de rango permitido")
        Else
            MessageBox.Show("Número dentro de rango")
        End If
    End Sub
End Class
```

Programa 10: Verificación de igualdad

Enunciado: Comprueba si dos textos ingresados son iguales usando operadores relacionales.

```
Public Class Form1
    Private Sub btnIguales_Click(sender As Object, e As EventArgs) Handles
btnIguales.Click
        Dim t1 As String = txtTexto1.Text
        Dim t2 As String = txtTexto2.Text
        If t1 = t2 Then
            MessageBox.Show("Los textos son iguales")
        Else
            MessageBox.Show("Los textos son diferentes")
        End If
    End Sub
End Class
```

Programa 11: Incremento y decremento

Enunciado: Incrementa y decrementa un número usando operadores `+=` y `-=`.

```
Public Class Form1

    Private Sub btnIncrementar_Click(sender As Object, e As EventArgs) Handles
btnIncrementar.Click

        Dim num As Integer = CInt(txtNumero.Text)

        num += 1

        MessageBox.Show("Después de incrementar: " & num)

        num -= 1

        MessageBox.Show("Después de decrementar: " & num)

    End Sub

End Class
```

Programa 12: Concatenación con +=

Enunciado: Añade texto adicional a una cadena usando operador de concatenación compuesto.

```
Public Class Form1

    Private Sub btnAgregarTexto_Click(sender As Object, e As EventArgs) Handles
btnAgregarTexto.Click

        Dim mensaje As String = txtMensaje.Text

        mensaje += " - VB.NET Form"

        MessageBox.Show("Texto final: " & mensaje)

    End Sub

End Class
```

Programa 13: Operador ternario (If inline)

Enunciado: Evalúa si un número es positivo o negativo usando expresión condicional inline (`If`).

```
Public Class Form1

    Private Sub btnTernario_Click(sender As Object, e As EventArgs) Handles
btnTernario.Click

        Dim num As Integer = CInt(txtNumero.Text)

        Dim resultado As String = If(num >= 0, "Positivo", "Negativo")

        MessageBox.Show("El número es: " & resultado)

    End Sub

End Class
```

Programa 14: Comparación de cadenas con <>

Enunciado: Comprueba si dos cadenas son diferentes y muestra el resultado.

```
Public Class Form1

    Private Sub btnDiferente_Click(sender As Object, e As EventArgs) Handles
btnDiferente.Click

        Dim c1 As String = txtCadena1.Text

        Dim c2 As String = txtCadena2.Text

        If c1 <> c2 Then

            MessageBox.Show("Las cadenas son diferentes")

        Else

            MessageBox.Show("Las cadenas son iguales")

        End If

    End Sub

End Class
```

Programa 15: Evaluación de rango con operadores lógicos

Enunciado: Verifica si un número está fuera del rango 1 a 50 usando `Not` y operadores lógicos.

```
Public Class Form1

    Private Sub btnRango_Click(sender As Object, e As EventArgs) Handles
btnRango.Click

        Dim num As Integer = CInt(txtNumero.Text)

        If Not (num >= 1 AndAlso num <= 50) Then

            MessageBox.Show("Número fuera de rango")

        Else

            MessageBox.Show("Número dentro de rango")

        End If

    End Sub

End Class
```

Programa 16: Operaciones combinadas

Enunciado: Realiza suma, resta y comparación en la misma expresión y muestra el resultado.

```
Public Class Form1

    Private Sub btnCombinado_Click(sender As Object, e As EventArgs) Handles
btnCombinado.Click

        Dim a As Integer = CInt(txtA.Text)

        Dim b As Integer = CInt(txtB.Text)

        Dim resultado As String = If(a + b > 10, "Mayor que 10", "Menor o igual
que 10")

        MessageBox.Show("Suma: " & (a + b) & " - " & resultado)

    End Sub

End Class
```

Programa 17: Validación de entrada numérica

Enunciado: Verifica si la entrada es numérica antes de realizar operaciones.

```
Public Class Form1

    Private Sub btnValidar_Click(sender As Object, e As EventArgs) Handles
btnValidar.Click

        Dim entrada As String = txtEntrada.Text

        If IsNumeric(entrada) Then

            Dim num As Double = CDbl(entrada)

            MessageBox.Show("Número ingresado: " & num)

        Else

            MessageBox.Show("Error: No es un número válido")

        End If

    End Sub

End Class
```

Programa 18: Comparación múltiple con `AndAlso` y `OrElse`

Enunciado: Evalúa si un número está entre 10 y 20 o es mayor que 50.

```
Public Class Form1

    Private Sub btnCondicional_Click(sender As Object, e As EventArgs) Handles
btnCondicional.Click

        Dim num As Integer = CInt(txtNumero.Text)

        If (num >= 10 AndAlso num <= 20) OrElse num > 50 Then

            MessageBox.Show("Número válido según condición")

        Else

            MessageBox.Show("Número no cumple condiciones")

        End If

    End Sub

End Class
```

Programa 19: Operador Not y comparación

Enunciado: Verifica si un checkbox no está marcado y muestra un mensaje.

```
Public Class Form1

    Private Sub btnVerificarCheck_Click(sender As Object, e As EventArgs)
Handles btnVerificarCheck.Click

        If Not chkAceptar.Checked Then

            MessageBox.Show("Debes aceptar los términos")

        Else

            MessageBox.Show("Términos aceptados")

        End If

    End Sub

End Class
```

Programa 20: Concatenación y operadores condicionales

Enunciado: Combina nombre y edad, y evalúa si es mayor o menor de edad en un solo mensaje.

```
Public Class Form1

    Private Sub btnInfo_Click(sender As Object, e As EventArgs) Handles
btnInfo.Click

        Dim nombre As String = txtNombre.Text

        Dim edad As Integer = CInt(txtEdad.Text)

        Dim mensaje As String = nombre & " - " & If(edad >= 18, "Mayor de edad",
"Menor de edad")

        MessageBox.Show(mensaje)

    End Sub

End Class
```

Ejercicios

Programa 1: Determinar si un número es positivo o negativo (If-Else)

Enunciado: El usuario introduce un número y el programa muestra si es positivo, negativo o cero.

```
Public Class Form1
    Private Sub btnVerificar_Click(sender As Object, e As EventArgs) Handles
btnVerificar.Click
        Dim numero As Integer = CInt(txtNumero.Text)
        If numero > 0 Then
            MessageBox.Show("El número es positivo")
        ElseIf numero < 0 Then
            MessageBox.Show("El número es negativo")
        Else
            MessageBox.Show("El número es cero")
        End If
    End Sub
End Class
```

Programa 2: Determinar día de la semana (Select Case)

Enunciado: Según un número del 1 al 7, mostrar el día de la semana correspondiente.

```
Public Class Form1
    Private Sub btnDia_Click(sender As Object, e As EventArgs) Handles
btnDia.Click
        Dim dia As Integer = CInt(txtDia.Text)
        Select Case dia
            Case 1 : MessageBox.Show("Lunes")
            Case 2 : MessageBox.Show("Martes")
            Case 3 : MessageBox.Show("Miércoles")
            Case 4 : MessageBox.Show("Jueves")
            Case 5 : MessageBox.Show("Viernes")
            Case 6 : MessageBox.Show("Sábado")
            Case 7 : MessageBox.Show("Domingo")
            Case Else : MessageBox.Show("Número inválido")
        End Select
    End Sub
End Class
```

Programa 3: Sumar los primeros N números (For)

Enunciado: Sumar todos los números desde 1 hasta N ingresado por el usuario.

```
Public Class Form1
    Private Sub btnSumar_Click(sender As Object, e As EventArgs) Handles
btnSumar.Click
        Dim n As Integer = CInt(txtN.Text)
        Dim suma As Integer = 0
        For i As Integer = 1 To n
            suma += i
        Next
        MessageBox.Show("La suma es " & suma)
    End Sub
End Class
```

Programa 4: Mostrar números pares hasta N (While)

Enunciado: Mostrar todos los números pares desde 2 hasta N.

```
Public Class Form1
    Private Sub btnPares_Click(sender As Object, e As EventArgs) Handles
btnPares.Click
        Dim n As Integer = CInt(txtN.Text)
        Dim i As Integer = 2
        While i <= n
            MessageBox.Show(i)
            i += 2
        End While
    End Sub
End Class
```

Programa 5: Contar hasta N (Do Until)

Enunciado: Contar desde 1 hasta N usando `Do Until`.

```
Public Class Form1
    Private Sub btnContar_Click(sender As Object, e As EventArgs) Handles
btnContar.Click
        Dim n As Integer = CInt(txtN.Text)
        Dim i As Integer = 1
        Do Until i > n
            MessageBox.Show(i)
            i += 1
        Loop
    End Sub
End Class
```

Programa 6: Verificar número divisible por 3 y 5 (If anidado)

Enunciado: Determinar si un número es divisible por 3, por 5 o por ambos.

```
Public Class Form1
    Private Sub btnDivisible_Click(sender As Object, e As EventArgs) Handles
btnDivisible.Click
        Dim numero As Integer = CInt(txtNumero.Text)
        If numero Mod 3 = 0 Then
            If numero Mod 5 = 0 Then
                MessageBox.Show("Divisible por 3 y 5")
            Else
                MessageBox.Show("Divisible por 3")
            End If
        ElseIf numero Mod 5 = 0 Then
            MessageBox.Show("Divisible por 5")
        Else
            MessageBox.Show("No divisible ni por 3 ni por 5")
        End If
    End Sub
End Class
```

Programa 7: Buscar un número en un rango (For + Exit For)

Enunciado: Buscar un número en un rango del 1 al 100 y detener el bucle cuando se encuentra.

```
Public Class Form1
    Private Sub btnBuscar_Click(sender As Object, e As EventArgs) Handles
btnBuscar.Click
        Dim buscar As Integer = CInt(txtNumero.Text)
        For i As Integer = 1 To 100
            If i = buscar Then
                MessageBox.Show("Número encontrado: " & i)
                Exit For
            End If
        Next
    End Sub
End Class
```

Programa 8: Saltar múltiplos de 4 (For + Continue For)

Enunciado: Mostrar los números del 1 al 20, pero saltar los múltiplos de 4.

```
Public Class Form1
    Private Sub btnMostrar_Click(sender As Object, e As EventArgs) Handles
btnMostrar.Click
        For i As Integer = 1 To 20
            If i Mod 4 = 0 Then
                Continue For
            End If
            MessageBox.Show(i)
        Next
    End Sub
End Class
```

Programa 9: Factorial de un número (While + Exit)

Enunciado: Calcular el factorial de un número, salir del bucle si el número es negativo.

```
Public Class Form1
    Private Sub btnFactorial_Click(sender As Object, e As EventArgs) Handles
btnFactorial.Click
        Dim n As Integer = CInt(txtNumero.Text)
        If n < 0 Then
            MessageBox.Show("Número negativo, no se puede calcular factorial")
            Return
        End If

        Dim factorial As Long = 1
        Dim i As Integer = 1
        While i <= n
            factorial *= i
            i += 1
        End While
        MessageBox.Show("Factorial: " & factorial)
    End Sub
End Class
```

Programa 10: Validar contraseña (If + Return)

Enunciado: Solicitar una contraseña y terminar la función si es incorrecta.

```
Public Class Form1
    Private Sub btnLogin_Click(sender As Object, e As EventArgs) Handles
btnLogin.Click
        Dim password As String = txtPassword.Text
        If password <> "12345" Then
            MessageBox.Show("Contraseña incorrecta")
            Return
        End If
        MessageBox.Show("Contraseña correcta, acceso permitido")
    End Sub
End Class
```

Programa 11: Sumar números pares hasta N (For + If)

Enunciado: Sumar solo los números pares hasta N.

```
Public Class Form1
    Private Sub btnSumar_Click(sender As Object, e As EventArgs) Handles
btnSumar.Click
        Dim n As Integer = CInt(txtN.Text)
        Dim suma As Integer = 0
        For i As Integer = 1 To n
            If i Mod 2 = 0 Then
                suma += i
            End If
        Next
        MessageBox.Show("Suma de pares: " & suma)
    End Sub
End Class
```

Programa 12: Mostrar la tabla de multiplicar (For anidado)

Enunciado: Mostrar la tabla de multiplicar del 1 al 5.

```
Public Class Form1
    Private Sub btnTabla_Click(sender As Object, e As EventArgs) Handles
btnTabla.Click
        For i As Integer = 1 To 5
            For j As Integer = 1 To 10
                MessageBox.Show(i & " x " & j & " = " & (i * j))
            Next
        Next
    End Sub
End Class
```

Programa 13: Número primo (For + If + Exit For)

Enunciado: Determinar si un número es primo.

```
Public Class Form1
    Private Sub btnPrimo_Click(sender As Object, e As EventArgs) Handles
btnPrimo.Click
        Dim n As Integer = CInt(txtNumero.Text)
        If n < 2 Then
            MessageBox.Show("No es primo")
            Return
        End If

        For i As Integer = 2 To n - 1
            If n Mod i = 0 Then
                MessageBox.Show("No es primo")
                Exit Sub
            End If
        Next
        MessageBox.Show("Es primo")
    End Sub
End Class
```

Programa 14: Contar números positivos y negativos (If + For)

Enunciado: Contar cuántos números positivos y negativos hay en un conjunto de 5 números.

```
Public Class Form1
    Private Sub btnContar_Click(sender As Object, e As EventArgs) Handles
btnContar.Click
        Dim positivos As Integer = 0
        Dim negativos As Integer = 0

        Dim numeros() As Integer = {CInt(txtNum1.Text), CInt(txtNum2.Text),
CInt(txtNum3.Text), CInt(txtNum4.Text), CInt(txtNum5.Text)}

        For Each num As Integer In numeros
            If num > 0 Then
                positivos += 1
            ElseIf num < 0 Then
                negativos += 1
            End If
        Next
        MessageBox.Show("Positivos: " & positivos & " Negativos: " & negativos)
    End Sub
End Class
```

Programa 15: Mostrar números hasta N (Do Until + If)

Enunciado: Mostrar números del 1 al N pero detenerse si llega a 7.

```
Public Class Form1
    Private Sub btnMostrar_Click(sender As Object, e As EventArgs) Handles
btnMostrar.Click
        Dim n As Integer = CInt(txtN.Text)
        Dim i As Integer = 1
        Do Until i > n
            If i = 7 Then
                Exit Do
            End If
            MessageBox.Show(i)
            i += 1
        Loop
    End Sub
End Class
```

Programa 16: Calcular promedio con validación (If + Return)

Enunciado: Calcular el promedio de 3 notas, validar que no sean mayores a 10.

```
Public Class Form1
    Private Sub btnPromedio_Click(sender As Object, e As EventArgs) Handles
btnPromedio.Click
        Dim notas() As Double = {CDbl(txtNota1.Text), CDbl(txtNota2.Text),
CDbl(txtNota3.Text)}

        For Each nota As Double In notas
            If nota > 10 Then
                MessageBox.Show("Nota inválida")
                Return
            End If
        Next

        Dim promedio As Double = (notas(0) + notas(1) + notas(2)) / 3
        MessageBox.Show("Promedio: " & promedio)
    End Sub
End Class
```

Programa 17: Sumar números hasta que se introduce cero (While + Exit While)

Enunciado: Sumar números ingresados hasta que se introduce un cero.

```
Public Class Form1
    Private Sub btnSumar_Click(sender As Object, e As EventArgs) Handles
btnSumar.Click
        Dim suma As Integer = 0
        Dim numero As Integer

        Do
            numero = CInt(InputBox("Ingrese un número (0 para terminar)"))
            If numero = 0 Then
                Exit Do
            End If
            suma += numero
        Loop
        MessageBox.Show("Suma total: " & suma)
    End Sub
End Class
```

Programa 18: Mostrar números impares con For + Continue

Enunciado: Mostrar del 1 al 20, saltando los números pares.

```
Public Class Form1
    Private Sub btnImpares_Click(sender As Object, e As EventArgs) Handles
btnImpares.Click
        For i As Integer = 1 To 20
            If i Mod 2 = 0 Then
                Continue For
            End If
            MessageBox.Show(i)
        Next
    End Sub
End Class
```

Programa 19: Clasificar edad (If + ElseIf)

Enunciado: Según la edad ingresada, clasificar como niño, adolescente, adulto o adulto mayor.

```
Public Class Form1
    Private Sub btnClasificar_Click(sender As Object, e As EventArgs) Handles
btnClasificar.Click
        Dim edad As Integer = CInt(txtEdad.Text)
        If edad < 13 Then
            MessageBox.Show("Niño")
        ElseIf edad < 20 Then
            MessageBox.Show("Adolescente")
        ElseIf edad < 60 Then
            MessageBox.Show("Adulto")
        Else
            MessageBox.Show("Adulto mayor")
        End If
    End Sub
End Class
```

Programa 20: Mostrar cuadrados de números (For)

Enunciado: Mostrar el cuadrado de los números del 1 al 10.

```
Public Class Form1
    Private Sub btnCuadrados_Click(sender As Object, e As EventArgs) Handles
btnCuadrados.Click
        For i As Integer = 1 To 10
            MessageBox.Show(i & "^2 = " & (i * i))
        Next
    End Sub
End Class
```

Parte II – Programación Orientada a Objetos

Capítulo 6. Introducción a la POO en VB.NET

La **Programación Orientada a Objetos (POO)** permite estructurar programas mediante **objetos** que combinan **datos** y **funciones**, facilitando la reutilización y organización del código. VB.NET es un lenguaje totalmente orientado a objetos, compatible con todas las características de la POO: **clases, objetos, encapsulación, herencia, polimorfismo y sobrecarga**.

6.1. Concepto de Clases y Objetos

- **Clase:** Plantilla o molde que define propiedades (atributos) y métodos (funciones) de un objeto.
- **Objeto:** Instancia concreta de una clase.

Ejemplo:

```
Public Class Persona
    ' Propiedades
    Public Nombre As String
    Public Edad As Integer

    ' Método
    Public Sub MostrarInfo()
        MessageBox.Show("Nombre: " & Nombre & ", Edad: " & Edad)
    End Sub
End Class
```

Uso en un formulario:

```
Private Sub btnCrearPersona_Click(sender As Object, e As EventArgs) Handles
btnCrearPersona.Click
    Dim p As New Persona()
    p.Nombre = "Ana"
    p.Edad = 25
    p.MostrarInfo()
End Sub
```

6.2. Constructores y Destructores

- **Constructor:** Método especial que se ejecuta al crear un objeto. Sirve para inicializar propiedades.
- **Destructor:** Método que se ejecuta cuando el objeto es eliminado, liberando recursos.

Ejemplo de constructor:

```
Public Class Persona
    Public Nombre As String
    Public Edad As Integer

    ' Constructor
    Public Sub New(nombre As String, edad As Integer)
        Me.Nombre = nombre
        Me.Edad = edad
    End Sub

    Public Sub MostrarInfo()
        MessageBox.Show("Nombre: " & Nombre & ", Edad: " & Edad)
    End Sub
End Class
```

Uso en Windows Forms:

```
Private Sub btnCrearPersona_Click(sender As Object, e As EventArgs) Handles
btnCrearPersona.Click
    Dim p As New Persona("Carlos", 30)
    p.MostrarInfo()
End Sub
```

Nota: En VB.NET no se usa habitualmente un destructor explícito; la recolección de basura (Garbage Collector) se encarga de liberar memoria.

6.3. Encapsulación, Herencia y Polimorfismo

Encapsulación

Permite proteger los datos de una clase usando **modificadores de acceso**: `Public`, `Private`, `Protected`.

```
Public Class CuentaBancaria
    Private saldo As Double

    Public Sub Depositar(monto As Double)
        saldo += monto
    End Sub

    Public Function ObtenerSaldo() As Double
        Return saldo
    End Function
End Class
```

Herencia

Permite crear una clase basada en otra, heredando sus propiedades y métodos.

```
Public Class Empleado
    Inherits Persona ' Hereda de Persona
    Public Puesto As String

    Public Sub MostrarEmpleado()
        MessageBox.Show("Nombre: " & Nombre & ", Puesto: " & Puesto)
    End Sub
End Class
```

Polimorfismo

Permite que un mismo método tenga comportamientos distintos según la clase.

```
Public Class Animal
    Public Overridable Sub HacerSonido()
        MessageBox.Show("Sonido genérico")
    End Sub
End Class

Public Class Perro
    Inherits Animal
    Public Overrides Sub HacerSonido()
        MessageBox.Show("Guau")
    End Sub
End Class
```

Uso en formulario:

```
Private Sub btnSonido_Click(sender As Object, e As EventArgs) Handles
btnSonido.Click
    Dim a As Animal = New Perro()
    a.HacerSonido() ' Muestra "Guau"
End Sub
```

6.4. Sobrecarga de métodos

La sobrecarga permite crear **varias versiones de un mismo método** con diferente número o tipo de parámetros.

```
Public Class Calculadora
    Public Function Sumar(a As Integer, b As Integer) As Integer
        Return a + b
    End Function

    Public Function Sumar(a As Double, b As Double) As Double
        Return a + b
    End Function

    Public Function Sumar(a As Integer, b As Integer, c As Integer) As Integer
        Return a + b + c
    End Function
End Class
```

Uso en formulario:

```
Private Sub btnSumar_Click(sender As Object, e As EventArgs) Handles
btnSumar.Click
    Dim calc As New Calculadora()
    MessageBox.Show("Suma 2 enteros: " & calc.Sumar(2, 3))
    MessageBox.Show("Suma 2 dobles: " & calc.Sumar(2.5, 3.7))
    MessageBox.Show("Suma 3 enteros: " & calc.Sumar(1, 2, 3))
End Sub
```

Capítulo 7 – Propiedades, Métodos y Eventos

En VB.NET, la programación orientada a objetos se fortalece mediante **propiedades, métodos y eventos**. Estos elementos permiten encapsular datos, definir comportamientos y reaccionar a acciones del usuario en Windows Forms.

7.1. Creación de Propiedades en Clases

Las **propiedades** permiten acceder y modificar los valores de los atributos de una clase de manera controlada. Se usan los bloques `Get` y `Set` para encapsular la lógica de lectura y escritura.

Ejemplo:

```
Public Class Persona
    Private _nombre As String
    Private _edad As Integer

    ' Propiedad Nombre
    Public Property Nombre() As String
        Get
            Return _nombre
        End Get
        Set(value As String)
            _nombre = value
        End Set
    End Property

    ' Propiedad Edad
    Public Property Edad() As Integer
        Get
            Return _edad
        End Get
        Set(value As Integer)
            If value >= 0 Then
                _edad = value
            Else
                MessageBox.Show("La edad no puede ser negativa")
            End If
        End Set
```

```
    End Property
End Class
```

Uso en Windows Forms:

```
Private Sub btnMostrar_Click(sender As Object, e As EventArgs) Handles
btnMostrar.Click
    Dim p As New Persona()
    p.Nombre = "Laura"
    p.Edad = 28
    MessageBox.Show("Nombre: " & p.Nombre & ", Edad: " & p.Edad)
End Sub
```

7.2. Métodos Públicos, Privados y Compartidos

Métodos Públicos y Privados

- **Públicos (`Public`)**: accesibles desde cualquier parte del programa.
- **Privados (`Private`)**: accesibles solo dentro de la clase.

```
Public Class Calculadora
    Private Function Multiplicar(a As Integer, b As Integer) As Integer
        Return a * b
    End Function

    Public Function Sumar(a As Integer, b As Integer) As Integer
        Return a + b
    End Function

    Public Sub MostrarMultiplicacion(a As Integer, b As Integer)
        MessageBox.Show("Multiplicación: " & Multiplicar(a, b))
    End Sub
End Class
```

Uso en formulario:

```
Private Sub btnOperaciones_Click(sender As Object, e As EventArgs) Handles
btnOperaciones.Click
    Dim calc As New Calculadora()
    MessageBox.Show("Suma: " & calc.Sumar(5, 3))
    calc.MostrarMultiplicacion(5, 3)
End Sub
```

Métodos Compartidos (`Shared`)

- No requieren instancia de la clase para ser usados.
- Se accede a ellos directamente con el nombre de la clase.

```
Public Class Utilidades
    Public Shared Function Cuadrado(x As Integer) As Integer
        Return x * x
    End Function
End Class
```

Uso en formulario:

```
Private Sub btnCuadrado_Click(sender As Object, e As EventArgs) Handles
btnCuadrado.Click
    MessageBox.Show("Cuadrado de 4: " & Utilidades.Cuadrado(4))
End Sub
```

7.3. Delegados y Eventos Personalizados

Delegados

Un **delegado** es un tipo que define la firma de un método que se puede asignar a variables, permitiendo ejecutar métodos de forma dinámica.

```
Public Delegate Sub MiDelegado(mensaje As String)
```

Eventos Personalizados

- Los eventos permiten que una clase **notifique** a otras clases que algo ocurrió.
- Se definen con `Event` y se disparan con `RaiseEvent`.

Ejemplo completo:

```
Public Class BotonEspecial
    ' Delegado
    Public Delegate Sub EventoClickEspecial(sender As Object, e As EventArgs)
    ' Evento
    Public Event ClickEspecial As EventoClickEspecial

    ' Método para disparar el evento
    Public Sub DispararClick()
        RaiseEvent ClickEspecial(Me, EventArgs.Empty)
    End Sub
End Class
```

Uso en formulario:

```
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    Dim btnEsp As New BotonEspecial()
    AddHandler btnEsp.ClickEspecial, AddressOf ManejarClickEspecial
    btnEsp.DispararClick()
End Sub

Private Sub ManejarClickEspecial(sender As Object, e As EventArgs)
    MessageBox.Show("Evento personalizado disparado")
End Sub
```

Ejercicios

---

1. Programa 1 – Crear una clase simple y mostrar sus propiedades

Enunciado: Este programa define una clase `Persona` con propiedades `Nombre` y `Edad`, crea un objeto y muestra sus valores en un `Label`.

```
Public Class Persona
    Public Property Nombre As String
    Public Property Edad As Integer
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim p As New Persona()
        p.Nombre = "Ana"
        p.Edad = 25
        MessageBox.Show("Nombre: " & p.Nombre & vbCrLf & "Edad: " & p.Edad)
    End Sub
End Class
```

---

2. Programa 2 – Constructor simple

Enunciado: Muestra cómo usar un constructor para inicializar un objeto al crearlo.

```
Public Class Persona
    Public Property Nombre As String
    Public Property Edad As Integer

    Public Sub New(n As String, e As Integer)
        Nombre = n
        Edad = e
    End Sub
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim p As New Persona("Luis", 30)
        MessageBox.Show("Nombre: " & p.Nombre & vbCrLf & "Edad: " & p.Edad)
    End Sub
End Class
```

---

3. Programa 3 – Destructor

Enunciado: Muestra cómo se define un destructor y cuándo se ejecuta (al liberar el objeto).

```
Public Class Persona
    Public Property Nombre As String

    Protected Overrides Sub Finalize()
        MessageBox.Show("El objeto Persona ha sido destruido")
        MyBase.Finalize()
    End Sub
End Class

Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim p As New Persona()
        p.Nombre = "Carlos"
        p = Nothing
        GC.Collect() ' Forzar recolección de basura
    End Sub
End Class
```

---

4. Programa 4 – Encapsulación

Enunciado: Uso de `Private` y `Public` para controlar el acceso a propiedades.

```
Public Class CuentaBancaria
    Private saldo As Double

    Public Sub Depositar(monto As Double)
        saldo += monto
    End Sub

    Public Function MostrarSaldo() As Double
        Return saldo
    End Function
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim cuenta As New CuentaBancaria()
        cuenta.Depositar(500)
        MessageBox.Show("Saldo actual: " & cuenta.MostrarSaldo())
    End Sub
End Class
```

---

5. Programa 5 – Herencia simple

Enunciado: Creación de una clase base `Animal` y una clase derivada `Perro`.

```
Public Class Animal
    Public Property Nombre As String
    Public Sub HacerSonido()
        MessageBox.Show("Sonido de animal")
    End Sub
End Class

Public Class Perro
    Inherits Animal
    Public Sub HacerSonidoPerro()
        MessageBox.Show("Guau guau!")
    End Sub
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim dog As New Perro()
        dog.Nombre = "Bobby"
        dog.HacerSonidoPerro()
    End Sub
End Class
```

---

6. Programa 6 – Polimorfismo con Overrides

Enunciado: Sobrescribir un método de la clase base para comportamiento diferente.

```
Public Class Animal
    Public Overridable Sub HacerSonido()
        MessageBox.Show("Sonido de animal")
    End Sub
End Class

Public Class Perro
    Inherits Animal
    Public Overrides Sub HacerSonido()
        MessageBox.Show("Guau guau!")
    End Sub
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim a As Animal = New Perro()
        a.HacerSonido() ' Llamará al método sobrescrito en Perro
    End Sub
End Class
```

---

7. Programa 7 – Sobrecarga de métodos

Enunciado: Definir varios métodos con el mismo nombre pero diferentes parámetros.

```
Public Class Calculadora
    Public Function Sumar(a As Integer, b As Integer) As Integer
        Return a + b
    End Function

    Public Function Sumar(a As Double, b As Double) As Double
        Return a + b
    End Function
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim calc As New Calculadora()
        MessageBox.Show("Enteros: " & calc.Sumar(3, 4) & vbCrLf & "Decimales: "
& calc.Sumar(2.5, 3.7))
    End Sub
End Class
```

---

8. Programa 8 – Propiedades con Get y Set

Enunciado: Demuestra cómo encapsular datos usando propiedades personalizadas.

```
Public Class Persona
    Private _edad As Integer
    Public Property Edad As Integer
        Get
            Return _edad
        End Get
        Set(value As Integer)
            If value >= 0 Then
                _edad = value
            Else
                MessageBox.Show("Edad no válida")
            End If
        End Set
    End Property
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim p As New Persona()
        p.Edad = 28
        MessageBox.Show("Edad: " & p.Edad)
    End Sub
End Class
```

---

9. Programa 9 – Constructor con múltiples parámetros

Enunciado: Inicializa varios atributos de la clase en un solo constructor.

```
Public Class Vehiculo
    Public Property Marca As String
    Public Property Modelo As String
    Public Property Año As Integer

    Public Sub New(m As String, mo As String, a As Integer)
        Marca = m
        Modelo = mo
        Año = a
    End Sub
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim car As New Vehiculo("Toyota", "Corolla", 2021)
        MessageBox.Show(car.Marca & " " & car.Modelo & " " & car.Año)
    End Sub
End Class
```

---

10. Programa 10 – Herencia con método compartido

Enunciado: Clase base con método, heredada por varias subclases.

```
Public Class Figura
    Public Overridable Sub Dibujar()
        MessageBox.Show("Dibujando figura")
    End Sub
End Class

Public Class Circulo
    Inherits Figura
    Public Overrides Sub Dibujar()
        MessageBox.Show("Dibujando círculo")
    End Sub
End Class

Public Class Cuadrado
    Inherits Figura
    Public Overrides Sub Dibujar()
        MessageBox.Show("Dibujando cuadrado")
    End Sub
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim f1 As Figura = New Circulo()
        Dim f2 As Figura = New Cuadrado()
        f1.Dibujar()
        f2.Dibujar()
    End Sub
End Class
```

---

11. Programa 11 – Objeto con método que devuelve valor

Enunciado: La clase tiene un método que devuelve un valor calculado y lo muestra en un `MessageBox`.

```
Public Class Rectangulo
    Public Property Base As Double
    Public Property Altura As Double

    Public Function CalcularArea() As Double
        Return Base * Altura
    End Function
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim r As New Rectangulo()
        r.Base = 5
        r.Altura = 3
        MessageBox.Show("Área del rectángulo: " & r.CalcularArea())
    End Sub
End Class
```

---

12. Programa 12 – Objeto con método que recibe parámetros

Enunciado: Método de clase que recibe parámetros y realiza operaciones con ellos.

```
Public Class Calculadora
    Public Function Multiplicar(a As Integer, b As Integer) As Integer
        Return a * b
    End Function
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim calc As New Calculadora()
        MessageBox.Show("3 x 4 = " & calc.Multiplicar(3, 4))
    End Sub
End Class
```

---

13. Programa 13 – Sobrecarga de métodos con distinto número de parámetros

Enunciado: Mostrar cómo se puede usar el mismo método con distinta cantidad de parámetros.

```
Public Class Calculadora
    Public Function Sumar(a As Integer, b As Integer) As Integer
        Return a + b
    End Function

    Public Function Sumar(a As Integer, b As Integer, c As Integer) As Integer
        Return a + b + c
    End Function
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim calc As New Calculadora()
        MessageBox.Show("2 números: " & calc.Sumar(2, 3) & vbCrLf & "3 números:
" & calc.Sumar(2, 3, 4))
    End Sub
End Class
```

---

14. Programa 14 – Propiedades con validación

Enunciado: Uso de `Set` para validar datos antes de asignarlos.

```
Public Class Empleado
    Private _salario As Double
    Public Property Salario As Double
        Get
            Return _salario
        End Get
        Set(value As Double)
            If value >= 0 Then
                _salario = value
            Else
                MessageBox.Show("Salario no puede ser negativo")
            End If
        End Set
    End Property
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim emp As New Empleado()
        emp.Salario = 1200
        MessageBox.Show("Salario: " & emp.Salario)
    End Sub
End Class
```

---

15. Programa 15 – Herencia con propiedades y métodos

Enunciado: Clase base con propiedades comunes y subclase que añade funcionalidades.

```
Public Class Persona
    Public Property Nombre As String
    Public Property Edad As Integer
End Class

Public Class Estudiante
    Inherits Persona
    Public Property Carrera As String
    Public Sub MostrarInfo()
        MessageBox.Show("Nombre: " & Nombre & vbCrLf & "Edad: " & Edad & vbCrLf
& "Carrera: " & Carrera)
    End Sub
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim est As New Estudiante()
        est.Nombre = "Marta"
        est.Edad = 20
        est.Carrera = "Ingeniería"
        est.MostrarInfo()
    End Sub
End Class
```

---

16. Programa 16 – Polimorfismo con interfaces

Enunciado: Implementación de polimorfismo mediante una interfaz.

```
Public Interface IAnimal
    Sub HacerSonido()
End Interface

Public Class Gato
    Implements IAnimal
    Public Sub HacerSonido() Implements IAnimal.HacerSonido
        MessageBox.Show("Miau miau")
    End Sub
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim animal As IAnimal = New Gato()
        animal.HacerSonido()
    End Sub
End Class
```

---

17. Programa 17 – Constructor predeterminado y constructor con parámetros

Enunciado: Demostración de sobrecarga de constructores.

```
Public Class Libro
    Public Property Titulo As String
    Public Property Autor As String

    Public Sub New()
        Titulo = "Desconocido"
        Autor = "Desconocido"
    End Sub

    Public Sub New(t As String, a As String)
        Titulo = t
        Autor = a
    End Sub
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim libro1 As New Libro()
        Dim libro2 As New Libro("1984", "Orwell")
        MessageBox.Show("Libro 1: " & libro1.Titulo & " - " & libro1.Autor &
vbCrLf &
                        "Libro 2: " & libro2.Titulo & " - " & libro2.Autor)
    End Sub
End Class
```

---

18. Programa 18 – Clase con método compartido (Shared)

Enunciado: Método `Shared` que se puede llamar sin crear un objeto.

```
Public Class Utilidades
    Public Shared Function Saludar(nombre As String) As String
        Return "Hola " & nombre
    End Function
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        MessageBox.Show(Utilidades.Saludar("Luis"))
    End Sub
End Class
```

---

19. Programa 19 – Polimorfismo con clase abstracta

Enunciado: Clase abstracta con método `MustOverride` y subclases que lo implementan.

```
Public MustInherit Class Figura
    Public MustOverride Function CalcularArea() As Double
End Class

Public Class Circulo
    Inherits Figura
    Public Property Radio As Double
    Public Overrides Function CalcularArea() As Double
        Return Math.PI * Radio * Radio
    End Function
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim c As New Circulo()
        c.Radio = 3
        MessageBox.Show("Área del círculo: " & c.CalcularArea())
    End Sub
End Class
```

---

20. Programa 20 – Sobrecarga de operadores (Opcional avanzado)

Enunciado: Demostración de cómo sobrecargar un operador `+` para sumar objetos.

```
Public Class Punto
    Public Property X As Integer
    Public Property Y As Integer

    Public Sub New(a As Integer, b As Integer)
        X = a
        Y = b
    End Sub

    Public Shared Operator +(p1 As Punto, p2 As Punto) As Punto
        Return New Punto(p1.X + p2.X, p1.Y + p2.Y)
    End Operator
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim p1 As New Punto(2, 3)
        Dim p2 As New Punto(4, 5)
        Dim p3 As Punto = p1 + p2
        MessageBox.Show("Punto resultante: (" & p3.X & ", " & p3.Y & ")")
    End Sub
End Class
```

Ejercicios

7.1 Creación de propiedades en clases

Programa 1 – Propiedad simple

Enunciado: Crear una clase `Persona` con una propiedad `Nombre`. Se asigna un valor y se muestra en pantalla.

```
Public Class Persona
    Public Property Nombre As String
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim p As New Persona()
        p.Nombre = "Ana"
        MessageBox.Show("Nombre: " & p.Nombre)
    End Sub
End Class
```

Programa 2 – Propiedad con validación

Enunciado: Propiedad `Edad` que valida que el valor no sea negativo.

```
Public Class Persona
    Private _edad As Integer
    Public Property Edad As Integer
        Get
            Return _edad
        End Get
        Set(value As Integer)
            If value >= 0 Then
                _edad = value
            Else
                MessageBox.Show("Edad no puede ser negativa")
            End If
        End Set
    End Property
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim p As New Persona()
        p.Edad = 25
        MessageBox.Show("Edad: " & p.Edad)
        p.Edad = -5  ' Mostrar validación
    End Sub
End Class
```

Programa 3 – Propiedad de solo lectura

Enunciado: Clase `Cuenta` con propiedad `Saldo` de solo lectura.

```
Public Class Cuenta
    Private _saldo As Double = 1000
    Public ReadOnly Property Saldo As Double
        Get
            Return _saldo
        End Get
    End Property
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim c As New Cuenta()
        MessageBox.Show("Saldo: " & c.Saldo)
    End Sub
End Class
```

Programa 4 – Propiedad calculada

Enunciado: Propiedad `Area` que calcula el área de un rectángulo.

```
Public Class Rectangulo
    Public Property Base As Double
    Public Property Altura As Double
    Public ReadOnly Property Area As Double
        Get
            Return Base * Altura
        End Get
    End Property
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim r As New Rectangulo() With {.Base = 5, .Altura = 3}
        MessageBox.Show("Área: " & r.Area)
    End Sub
End Class
```

Programa 5 – Propiedad compartida (Shared)

Enunciado: Propiedad `Contador` compartida entre todas las instancias de la clase `Usuario`.

```
Public Class Usuario
    Public Shared Contador As Integer = 0
    Public Sub New()
        Contador += 1
    End Sub
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim u1 As New Usuario()
        Dim u2 As New Usuario()
        MessageBox.Show("Usuarios creados: " & Usuario.Contador)
    End Sub
End Class
```

7.2 Métodos públicos, privados y compartidos (Shared)

Programa 6 – Método público

Enunciado: Método público que suma dos números.

```
Public Class Calculadora
    Public Function Sumar(a As Integer, b As Integer) As Integer
        Return a + b
    End Function
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim calc As New Calculadora()
        MessageBox.Show("Suma: " & calc.Sumar(3, 4))
    End Sub
End Class
```

Programa 7 – Método privado

Enunciado: Método privado dentro de una clase, accesible solo desde la propia clase.

```
Public Class Calculadora
    Private Function Restar(a As Integer, b As Integer) As Integer
        Return a - b
    End Function

    Public Function MostrarResta(a As Integer, b As Integer) As String
        Return "Resultado: " & Restar(a, b)
    End Function
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim calc As New Calculadora()
        MessageBox.Show(calc.MostrarResta(10, 3))
    End Sub
End Class
```

Programa 8 – Método compartido (Shared)

Enunciado: Método que se puede llamar sin instanciar la clase.

```
Public Class Utilidades
    Public Shared Function Saludar(nombre As String) As String
        Return "Hola " & nombre
    End Function
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        MessageBox.Show(Utilidades.Saludar("Luis"))
    End Sub
End Class
```

Programa 9 – Método con parámetros opcionales

Enunciado: Método público con parámetro opcional.

```
Public Class Calculadora
    Public Function Multiplicar(a As Integer, Optional b As Integer = 2) As
Integer
        Return a * b
    End Function
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim calc As New Calculadora()
        MessageBox.Show("Resultado: " & calc.Multiplicar(5)) ' b = 2 por defecto
    End Sub
End Class
```

Programa 10 – Método que devuelve valor

Enunciado: Método que calcula el área de un círculo y devuelve el resultado.

```
Public Class Circulo
    Public Property Radio As Double
    Public Function CalcularArea() As Double
        Return Math.PI * Radio * Radio
    End Function
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim c As New Circulo() With {.Radio = 3}
        MessageBox.Show("Área: " & c.CalcularArea())
    End Sub
End Class
```

7.3 Delegados y eventos personalizados

Programa 11 – Delegado básico

Enunciado: Crear un delegado que apunta a un método que muestra un mensaje.

```
Public Delegate Sub MiDelegado(mensaje As String)

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim del As MiDelegado = AddressOf MostrarMensaje
        del("Hola desde el delegado")
    End Sub

    Private Sub MostrarMensaje(texto As String)
        MessageBox.Show(texto)
    End Sub
End Class
```

Programa 12 – Evento personalizado

Enunciado: Crear una clase con un evento personalizado y dispararlo.

```
Public Class Alarma
    Public Event Activada(mensaje As String)

    Public Sub Activar()
        RaiseEvent Activada("¡Alarma activada!")
    End Sub
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim a As New Alarma()
        AddHandler a.Activada, AddressOf MostrarAlarma
        a.Activar()
    End Sub

    Private Sub MostrarAlarma(msg As String)
        MessageBox.Show(msg)
    End Sub
End Class
```

Programa 13 – Evento con parámetro de clase

Enunciado: Evento que pasa un objeto como argumento.

```
Public Class Persona
    Public Property Nombre As String
    Public Event Cambiado(sender As Persona)

    Public Sub CambiarNombre(nuevo As String)
        Nombre = nuevo
        RaiseEvent Cambiado(Me)
    End Sub
End Class

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim p As New Persona()
        AddHandler p.Cambiado, AddressOf NombreCambiado
        p.CambiarNombre("Carlos")
    End Sub

    Private Sub NombreCambiado(p As Persona)
        MessageBox.Show("Nombre actualizado a: " & p.Nombre)
    End Sub
End Class
```

Parte III – Windows Forms y Controles Básicos

Windows Forms es la plataforma de VB.NET para crear **interfaces gráficas de usuario (GUI)** en aplicaciones de escritorio. Permite manejar formularios, controles y eventos de manera intuitiva.

---

8. Introducción a Windows Forms

Un **Windows Form** es una ventana o cuadro de diálogo donde se colocan controles (botones, cuadros de texto, etiquetas, etc.) para interactuar con el usuario.

8.1. Crear un Formulario Básico

1. **Abrir Visual Studio** y crear un nuevo proyecto de tipo **Windows Forms App (.NET Framework o .NET Core)**.
2. Aparecerá un formulario prediseñado llamado `Form1`.
3. Puedes cambiar su **propiedad `Text`** para modificar el título de la ventana.
4. Para agregar controles, usa la **Caja de Herramientas (Toolbox)** y arrastra botones, etiquetas, cuadros de texto, etc.

Ejemplo de código para inicializar un formulario:

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Me.Text = "Mi Primer Formulario"
        Me.Size = New Size(400, 300) ' Ancho y alto del formulario
    End Sub
End Class
```

8.2. Ciclo de Vida de un Formulario

El ciclo de vida describe las **etapas que atraviesa un formulario desde su creación hasta su cierre**:

1. **Instanciación**: Se crea un objeto del formulario (`Dim frm As New Form1`).
2. **Carga (`Load`)**: Evento donde se inicializan valores y controles.
3. **Mostrar (`Show` o `ShowDialog`)**: El formulario se muestra en pantalla.
4. **Eventos de usuario**: Se manejan interacciones como clics de botón o cambios de texto.
5. **Cierre (`Closing` / `Closed`)**: Se liberan recursos y se termina la ejecución del formulario.

Ejemplo de manejo de eventos de ciclo de vida:

```
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    MessageBox.Show("Formulario cargado")
```

```
End Sub

Private Sub Form1_FormClosing(sender As Object, e As FormClosingEventArgs)
Handles MyBase.FormClosing
    Dim respuesta As DialogResult
    respuesta = MessageBox.Show("¿Desea cerrar el formulario?", "Cerrar",
MessageBoxButtons.YesNo)
    If respuesta = DialogResult.No Then
        e.Cancel = True
    End If
End Sub
```

8.3. Propiedades Principales de los Formularios

Algunas propiedades importantes que todo formulario debe conocer:

Propiedad	Descripción
`Text`	Título del formulario.
`Size`	Tamaño (ancho y alto) del formulario.
`BackColor`	Color de fondo del formulario.
`StartPosition`	Posición inicial del formulario en la pantalla (`CenterScreen`, `Manual`, etc.).
`FormBorderStyle`	Estilo de borde del formulario (`FixedSingle`, `Sizable`, etc.).
`MaximizeBox` / `MinimizeBox`	Habilitar o deshabilitar botones de maximizar/minimizar.
`ControlBox`	Muestra u oculta el control de cierre en la esquina superior.

Ejemplo práctico:

```
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    Me.Text = "Formulario Personalizado"
    Me.BackColor = Color.LightBlue
    Me.StartPosition = FormStartPosition.CenterScreen
    Me.FormBorderStyle = FormBorderStyle.FixedDialog
    Me.MaximizeBox = False
End Sub
```

---

9. Controles Básicos en Windows Forms

Los **controles** son los elementos gráficos que se colocan en los formularios para interactuar con el usuario. Cada control tiene propiedades, métodos y eventos propios.

9.1. Label

El `Label` se usa para mostrar texto estático en el formulario.

Propiedades importantes:

- `Text`: Texto que se muestra.
- `Font`: Fuente del texto.
- `ForeColor`: Color del texto.
- `BackColor`: Color de fondo.

Ejemplo:

```
Dim lblSaludo As New Label()
lblSaludo.Text = "Bienvenido a VB.NET"
lblSaludo.Font = New Font("Arial", 12, FontStyle.Bold)
lblSaludo.ForeColor = Color.DarkBlue
lblSaludo.Location = New Point(50, 30)
Me.Controls.Add(lblSaludo)
```

9.2. TextBox

El `TextBox` permite al usuario **ingresar texto**.

Propiedades importantes:

- `Text`: Texto contenido en el control.
- `MaxLength`: Máximo número de caracteres.
- `Multiline`: Permite varias líneas de texto.
- `ReadOnly`: Solo lectura si es `True`.

Ejemplo:

```
Dim txtNombre As New TextBox()
txtNombre.Location = New Point(50, 70)
txtNombre.Width = 200
txtNombre.MaxLength = 50
Me.Controls.Add(txtNombre)
```

9.3. Button

El `Button` permite ejecutar **acciones al hacer clic**.

Evento principal:

- `Click`: Se ejecuta cuando el usuario hace clic.

Ejemplo:

```
Dim btnSaludar As New Button()
btnSaludar.Text = "Saludar"
btnSaludar.Location = New Point(50, 110)
AddHandler btnSaludar.Click, AddressOf Me.BtnSaludar_Click
Me.Controls.Add(btnSaludar)

Private Sub BtnSaludar_Click(sender As Object, e As EventArgs)
    MessageBox.Show("Hola " & txtNombre.Text)
End Sub
```

9.4. CheckBox y RadioButton

CheckBox: Permite seleccionar **opciones independientes**.
RadioButton: Permite seleccionar **una opción de un grupo**.

Ejemplo CheckBox:

```
Dim chkAceptar As New CheckBox()
chkAceptar.Text = "Aceptar términos"
chkAceptar.Location = New Point(50, 150)
Me.Controls.Add(chkAceptar)
```

Ejemplo RadioButton:

```
Dim rdoSi As New RadioButton()
rdoSi.Text = "Sí"
rdoSi.Location = New Point(50, 180)
Dim rdoNo As New RadioButton()
rdoNo.Text = "No"
rdoNo.Location = New Point(120, 180)
Me.Controls.Add(rdoSi)
Me.Controls.Add(rdoNo)
```

9.5. ListBox y ComboBox

ListBox: Muestra una lista de elementos para selección múltiple o simple.
ComboBox: Lista desplegable, permite seleccionar un solo elemento o escribir uno nuevo.

Ejemplo ListBox:

```
Dim lstColores As New ListBox()
lstColores.Items.AddRange(New String() {"Rojo", "Verde", "Azul"})
lstColores.Location = New Point(50, 220)
Me.Controls.Add(lstColores)
```

Ejemplo ComboBox:

```
Dim cmbColores As New ComboBox()
cmbColores.Items.AddRange(New String() {"Rojo", "Verde", "Azul"})
cmbColores.Location = New Point(50, 260)
Me.Controls.Add(cmbColores)
```

9.6. PictureBox

El `PictureBox` muestra **imágenes** en el formulario.

Propiedades importantes:

- `Image`: Imagen a mostrar.
- `SizeMode`: Ajuste de imagen (`StretchImage`, `Zoom`, `Normal`).

Ejemplo:

```
Dim picLogo As New PictureBox()
picLogo.Image = Image.FromFile("C:\ruta\logo.png")
picLogo.Location = New Point(50, 300)
picLogo.Size = New Size(100, 100)
picLogo.SizeMode = PictureBoxSizeMode.StretchImage
Me.Controls.Add(picLogo)
```

9.7. GroupBox y Panel

Ambos sirven para **agrupar controles**:

- **GroupBox**: Muestra un marco con un título.
- **Panel**: Contenedor flexible sin título, permite scroll si hay muchos controles.

Ejemplo GroupBox:

```
Dim grpOpciones As New GroupBox()
grpOpciones.Text = "Opciones"
grpOpciones.Location = New Point(200, 70)
grpOpciones.Size = New Size(200, 150)

Dim chk1 As New CheckBox()
chk1.Text = "Opción 1"
chk1.Location = New Point(10, 20)
grpOpciones.Controls.Add(chk1)

Dim chk2 As New CheckBox()
chk2.Text = "Opción 2"
chk2.Location = New Point(10, 50)
grpOpciones.Controls.Add(chk2)

Me.Controls.Add(grpOpciones)
```

Ejemplo Panel:

```
Dim pnlContenedor As New Panel()
pnlContenedor.Location = New Point(200, 230)
pnlContenedor.Size = New Size(200, 100)
pnlContenedor.BackColor = Color.LightGray

Dim btnPanel As New Button()
btnPanel.Text = "Botón en Panel"
btnPanel.Location = New Point(10, 10)
pnlContenedor.Controls.Add(btnPanel)

Me.Controls.Add(pnlContenedor)
```

---

10. Propiedades de Controles y Formularios

Las **propiedades** definen el comportamiento, apariencia y posición de los **formularios y controles** en Windows Forms. Conocerlas permite crear interfaces más claras y funcionales.

10.1. Propiedades comunes: Name, Text, Enabled, Visible

Estas propiedades están presentes en casi todos los controles:

- **Name**: Identificador del control, usado para referenciarlo en código.
- **Text**: Texto que se muestra en el control.
- **Enabled**: Permite habilitar o deshabilitar el control (`True` o `False`).
- **Visible**: Permite mostrar u ocultar el control (`True` o `False`).

Ejemplo:

```
Dim btnEnviar As New Button()
btnEnviar.Name = "btnEnviar"
btnEnviar.Text = "Enviar"
btnEnviar.Enabled = True
btnEnviar.Visible = True
btnEnviar.Location = New Point(50, 50)
Me.Controls.Add(btnEnviar)
```

10.2. Propiedades de diseño: Font, BackColor, ForeColor

Estas propiedades controlan la **apariencia visual** del control:

- **Font**: Tipo, tamaño y estilo de la fuente.
- **BackColor**: Color de fondo del control.
- **ForeColor**: Color del texto o elementos visibles.

Ejemplo:

```
Dim lblTitulo As New Label()
lblTitulo.Text = "Formulario de Registro"
lblTitulo.Font = New Font("Arial", 14, FontStyle.Bold)
lblTitulo.ForeColor = Color.White
lblTitulo.BackColor = Color.DarkBlue
lblTitulo.AutoSize = True
lblTitulo.Location = New Point(50, 10)
Me.Controls.Add(lblTitulo)
```

10.3. Propiedades de disposición: Size, Location, Dock, Anchor

Estas propiedades definen la **posición y tamaño** del control dentro del formulario:

- **Size**: Tamaño del control (`Width` y `Height`).
- **Location**: Posición en coordenadas X e Y dentro del formulario.
- **Dock**: Ajusta el control a un borde del formulario (`Top`, `Bottom`, `Left`, `Right`, `Fill`).
- **Anchor**: Permite que el control se redimensione automáticamente al cambiar el tamaño del formulario.

Ejemplo:

```
Dim txtComentario As New TextBox()

txtComentario.Multiline = True

txtComentario.Size = New Size(300, 100)

txtComentario.Location = New Point(50, 100)

txtComentario.Anchor = AnchorStyles.Top Or AnchorStyles.Left Or
AnchorStyles.Right

Me.Controls.Add(txtComentario)
```

10.4. Métodos comunes: Focus, Select, Refresh

Estos métodos permiten **interactuar con los controles mediante código**:

- **Focus()**: Pone el cursor en el control, útil para cajas de texto.
- **Select()**: Selecciona el control, similar a `Focus()`.
- **Refresh()**: Fuerza al control a redibujarse, útil cuando se actualiza su apariencia dinámicamente.

Ejemplo:

```
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

    txtComentario.Focus()    ' Pone el cursor en el TextBox al iniciar

End Sub


Private Sub btnActualizar_Click(sender As Object, e As EventArgs) Handles
btnActualizar.Click

    lblTitulo.Text = "Contenido actualizado"

    lblTitulo.Refresh()      ' Redibuja el Label inmediatamente

End Sub
```

11.1. Modelo de eventos en VB.NET

El **modelo de eventos** en VB.NET se basa en la relación **evento → manejador de evento**:

- Un **evento** es algo que sucede en un control (por ejemplo, un clic en un botón).
- Un **manejador de eventos** es un procedimiento que se ejecuta cuando ocurre ese evento.

Ejemplo de un manejador de evento:

```
Private Sub btnEnviar_Click(sender As Object, e As EventArgs) Handles
btnEnviar.Click

    MessageBox.Show("Se presionó el botón Enviar")

End Sub
```

- `Handles btnEnviar.Click` indica que este procedimiento manejará el evento `Click` del botón `btnEnviar`.
- `sender` es el control que generó el evento.
- `e` contiene información adicional del evento.

11.2. Eventos de ratón

Los controles pueden responder a distintos eventos del ratón:

- **Click**: Se ejecuta al hacer clic simple.
- **DoubleClick**: Se ejecuta al hacer doble clic.
- **MouseDown**: Cuando se presiona un botón del ratón.
- **MouseUp**: Cuando se suelta un botón del ratón.
- **MouseMove**: Cuando el ratón se mueve sobre el control.

Ejemplo:

```
Private Sub lblEjemplo_MouseMove(sender As Object, e As MouseEventArgs) Handles
lblEjemplo.MouseMove

    lblEjemplo.Text = $"Posición del ratón: X={e.X}, Y={e.Y}"

End Sub
```

11.3. Eventos de teclado

Los controles pueden capturar la interacción del teclado:

- **KeyDown**: Cuando se presiona una tecla.
- **KeyUp**: Cuando se suelta una tecla.
- **KeyPress**: Cuando se presiona una tecla que genera un carácter.

Ejemplo:

```
Private Sub txtNombre_KeyPress(sender As Object, e As KeyPressEventArgs) Handles
txtNombre.KeyPress
    If Not Char.IsLetter(e.KeyChar) And Not Char.IsControl(e.KeyChar) Then
        e.Handled = True   ' Bloquea caracteres que no sean letras
        MessageBox.Show("Solo se permiten letras")
    End If
End Sub
```

11.4. Eventos de carga, activación y cierre de formularios

- **Load**: Se ejecuta al cargar el formulario.
- **Activated**: Se ejecuta cuando el formulario se activa (recibe el foco).
- **FormClosing**: Se ejecuta cuando el formulario se va a cerrar.
- **FormClosed**: Se ejecuta después de cerrar el formulario.

Ejemplo:

```
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    MessageBox.Show("Formulario cargado")
End Sub


Private Sub Form1_FormClosing(sender As Object, e As FormClosingEventArgs)
Handles Me.FormClosing
    If MessageBox.Show("¿Desea salir?", "Cerrar", MessageBoxButtons.YesNo) =
DialogResult.No Then
        e.Cancel = True   ' Cancela el cierre
    End If
End Sub
```

11.5. Crear y manejar eventos personalizados

Podemos definir **eventos propios** dentro de una clase y dispararlos según las necesidades del programa.

Ejemplo:

```
Public Class Contador
    Public Event ContadorAlcanzado(valor As Integer)

    Private _contador As Integer = 0

    Public Sub Incrementar()
        _contador += 1
        If _contador >= 10 Then
            RaiseEvent ContadorAlcanzado(_contador)  ' Dispara el evento
        End If
    End Sub
End Class
```

Uso del evento en un formulario:

```
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    Dim c As New Contador()
    AddHandler c.ContadorAlcanzado, AddressOf MostrarMensaje
    For i As Integer = 1 To 12
        c.Incrementar()
    Next
End Sub

Private Sub MostrarMensaje(valor As Integer)
    MessageBox.Show($"Contador alcanzó el valor {valor}")
End Sub
```

Resumen del Capítulo 11:

- Los eventos permiten **responder a acciones del usuario**.
- Los controles poseen eventos de **ratón, teclado y formulario**.
- Podemos crear **eventos personalizados** para diseñar aplicaciones más dinámicas y modulares.
- `AddHandler` y `RaiseEvent` son clave para trabajar con eventos propios.

Ejercicios

Arte III – Windows Forms y Controles Básicos

Tema: Introducción a Windows Forms

Programa 1 – Crear un formulario básico

Enunciado: Este programa crea un formulario vacío con título personalizado y tamaño definido.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Me.Text = "Formulario Básico"
        Me.Width = 400
        Me.Height = 300
    End Sub
End Class
```

Programa 2 – Cambiar color de fondo del formulario

Enunciado: Al cargar el formulario, se cambia su color de fondo.

```
Public Class Form2
    Private Sub Form2_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Me.BackColor = Color.LightBlue
    End Sub
End Class
```

Programa 3 – Centrar formulario en pantalla

Enunciado: Este formulario se centra automáticamente al abrirlo.

```
Public Class Form3
    Private Sub Form3_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Me.StartPosition = FormStartPosition.CenterScreen
    End Sub
End Class
```

Programa 4 – Cambiar tamaño mínimo y máximo del formulario

Enunciado: Limita el tamaño que puede tomar el formulario.

```
Public Class Form4
    Private Sub Form4_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Me.MinimumSize = New Size(300, 200)
        Me.MaximumSize = New Size(800, 600)
    End Sub
End Class
```

Programa 5 – Mostrar mensaje al cerrar el formulario

Enunciado: Captura el evento de cierre y muestra un mensaje de advertencia.

```
Public Class Form5
    Private Sub Form5_FormClosing(sender As Object, e As FormClosingEventArgs)
Handles MyBase.FormClosing
        MessageBox.Show("Está a punto de cerrar el formulario")
    End Sub
End Class
```

Programa 6 – Cambiar el icono del formulario

Enunciado: Se asigna un icono personalizado al formulario.

```
Public Class Form6
    Private Sub Form6_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Me.Icon = New Icon("C:\icono.ico")
    End Sub
End Class
```

Programa 7 – Mostrar formulario maximizado

Enunciado: Al iniciar, el formulario se abre en modo maximizado.

```
Public Class Form7
    Private Sub Form7_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Me.WindowState = FormWindowState.Maximized
    End Sub
End Class
```

Programa 8 – Mostrar formulario minimizado

Enunciado: Al iniciar, el formulario se abre en modo minimizado.

```
Public Class Form8
    Private Sub Form8_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Me.WindowState = FormWindowState.Minimized
    End Sub
End Class
```

Programa 9 – Cambiar título dinámicamente

Enunciado: El título del formulario se cambia con un botón.

```
Public Class Form9
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Me.Text = "Nuevo Título"
    End Sub
End Class
```

Programa 10 – Mostrar formulario con transparencia

Enunciado: El formulario se muestra semitransparente.

```
Public Class Form10
    Private Sub Form10_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        Me.Opacity = 0.7
    End Sub
End Class
```

Programa 11 – Cambiar borde del formulario

Enunciado: Se asigna un borde fijo al formulario para que no pueda redimensionarse.

```
Public Class Form11
    Private Sub Form11_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        Me.FormBorderStyle = FormBorderStyle.FixedDialog
    End Sub
End Class
```

Programa 12 – Cambiar estilo del borde del formulario

Enunciado: Cambia el borde a estilo resizable y muestra botones de minimizar/maximizar.

```
Public Class Form12
    Private Sub Form12_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        Me.FormBorderStyle = FormBorderStyle.Sizable
        Me.MinimizeBox = True
        Me.MaximizeBox = True
    End Sub
End Class
```

Programa 13 – Mostrar mensaje al cargar el formulario

Enunciado: Captura el evento `Load` y muestra un mensaje.

```
Public Class Form13
    Private Sub Form13_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        MessageBox.Show("Formulario cargado correctamente")
    End Sub
End Class
```

Programa 14 – Detectar formulario activo

Enunciado: Detecta cuando el formulario se activa y muestra un mensaje.

```
Public Class Form14
    Private Sub Form14_Activated(sender As Object, e As EventArgs) Handles
MyBase.Activated
        MessageBox.Show("Formulario activo")
    End Sub
End Class
```

Programa 15 – Detectar formulario inactivo

Enunciado: Detecta cuando el formulario pierde el foco.

```
Public Class Form15
    Private Sub Form15_Deactivate(sender As Object, e As EventArgs) Handles
MyBase.Deactivate
        MessageBox.Show("Formulario inactivo")
    End Sub
End Class
```

Programa 16 – Mostrar formulario con control de scroll

Enunciado: Habilita scroll vertical si el contenido excede el tamaño del formulario.

```
Public Class Form16
    Private Sub Form16_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        Me.AutoScroll = True
    End Sub
End Class
```

Programa 17 – Cambiar tamaño dinámicamente

Enunciado: Cambia el tamaño del formulario con un botón.

```
Public Class Form17
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Me.Size = New Size(600, 400)
    End Sub
End Class
```

Programa 18 – Mostrar formulario siempre encima

Enunciado: El formulario se mantiene siempre sobre otras ventanas.

```
Public Class Form18
    Private Sub Form18_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        Me.TopMost = True
    End Sub
End Class
```

Programa 19 – Cambiar icono y título dinámicamente

Enunciado: Cambia icono y título mediante un botón.

```
Public Class Form19
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Me.Text = "Formulario Actualizado"
        Me.Icon = New Icon("C:\nuevo_icono.ico")
    End Sub
End Class
```

Programa 20 – Combinar propiedades principales

Enunciado: Configura título, tamaño, color y posición del formulario al cargar.

```
Public Class Form20
    Private Sub Form20_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        Me.Text = "Formulario Combinado"
        Me.Size = New Size(500, 350)
        Me.BackColor = Color.LightGreen
        Me.StartPosition = FormStartPosition.CenterScreen
    End Sub
End Class
```

Ejercicios

Programa 1 – Label básico

Enunciado: Muestra un texto estático en el formulario mediante un `Label`.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim lbl As New Label()
        lbl.Text = "Hola, soy un Label"
        lbl.Location = New Point(50, 50)
        Me.Controls.Add(lbl)
    End Sub
End Class
```

Programa 2 – Label con propiedades

Enunciado: Cambia el color, fuente y tamaño del `Label`.

```
Public Class Form2
    Private Sub Form2_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim lbl As New Label()
        lbl.Text = "Label con estilo"
        lbl.Font = New Font("Arial", 14, FontStyle.Bold)
        lbl.ForeColor = Color.Red
        lbl.Location = New Point(50, 50)
        Me.Controls.Add(lbl)
    End Sub
End Class
```

Programa 3 – TextBox básico

Enunciado: Permite al usuario escribir texto en el formulario.

```
Public Class Form3
    Private Sub Form3_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim txt As New TextBox()
        txt.Location = New Point(50, 50)
        Me.Controls.Add(txt)
    End Sub
End Class
```

Programa 4 – TextBox multilinea

Enunciado: Crea un `TextBox` que acepta varias líneas de texto.

```
Public Class Form4
    Private Sub Form4_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim txt As New TextBox()
        txt.Multiline = True
        txt.Size = New Size(200, 100)
        txt.Location = New Point(50, 50)
        Me.Controls.Add(txt)
    End Sub
End Class
```

Programa 5 – Button básico

Enunciado: Muestra un botón que al hacer clic muestra un mensaje.

```
Public Class Form5
    Private Sub Form5_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim btn As New Button()
        btn.Text = "Haz clic"
        btn.Location = New Point(50, 50)
        AddHandler btn.Click, AddressOf Me.Button_Click
        Me.Controls.Add(btn)
    End Sub

    Private Sub Button_Click(sender As Object, e As EventArgs)
        MessageBox.Show("Botón clickeado")
    End Sub
End Class
```

Programa 6 – CheckBox básico

Enunciado: Permite seleccionar o deseleccionar una opción.

```
Public Class Form6
    Private Sub Form6_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim chk As New CheckBox()
        chk.Text = "Aceptar términos"
        chk.Location = New Point(50, 50)
        Me.Controls.Add(chk)
    End Sub
End Class
```

Programa 7 – CheckBox con evento

Enunciado: Muestra un mensaje cuando se selecciona o deselecciona el `CheckBox`.

```
Public Class Form7
    Private Sub Form7_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim chk As New CheckBox()
        chk.Text = "Habilitar opción"
        chk.Location = New Point(50, 50)
        AddHandler chk.CheckedChanged, AddressOf Me.Chk_CheckedChanged
        Me.Controls.Add(chk)
    End Sub

    Private Sub Chk_CheckedChanged(sender As Object, e As EventArgs)
        Dim chk As CheckBox = CType(sender, CheckBox)
        If chk.Checked Then
            MessageBox.Show("Opción habilitada")
        Else
            MessageBox.Show("Opción deshabilitada")
        End If
    End Sub
End Class
```

Programa 8 – RadioButton básico

Enunciado: Permite elegir entre varias opciones.

```
Public Class Form8
    Private Sub Form8_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim rb1 As New RadioButton()
        rb1.Text = "Opción 1"
        rb1.Location = New Point(50, 50)
        Me.Controls.Add(rb1)

        Dim rb2 As New RadioButton()
        rb2.Text = "Opción 2"
        rb2.Location = New Point(50, 80)
        Me.Controls.Add(rb2)
    End Sub
End Class
```

Programa 9 – RadioButton con evento

Enunciado: Muestra un mensaje según la opción seleccionada.

```
Public Class Form9
    Private Sub Form9_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim rb1 As New RadioButton()
        rb1.Text = "Opción A"
        rb1.Location = New Point(50, 50)
        AddHandler rb1.CheckedChanged, AddressOf Me.Rb_CheckedChanged
        Me.Controls.Add(rb1)

        Dim rb2 As New RadioButton()
        rb2.Text = "Opción B"
        rb2.Location = New Point(50, 80)
        AddHandler rb2.CheckedChanged, AddressOf Me.Rb_CheckedChanged
        Me.Controls.Add(rb2)
    End Sub

    Private Sub Rb_CheckedChanged(sender As Object, e As EventArgs)
        Dim rb As RadioButton = CType(sender, RadioButton)
        If rb.Checked Then
            MessageBox.Show("Seleccionaste " & rb.Text)
        End If
    End Sub
End Class
```

Programa 10 – ListBox básico

Enunciado: Muestra una lista de elementos que se pueden seleccionar.

```
Public Class Form10
    Private Sub Form10_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        Dim lst As New ListBox()
        lst.Items.Add("Elemento 1")
        lst.Items.Add("Elemento 2")
        lst.Items.Add("Elemento 3")
        lst.Location = New Point(50, 50)
        Me.Controls.Add(lst)
    End Sub
End Class
```

Programa 11 – ListBox con evento

Enunciado: Muestra un mensaje cuando se selecciona un elemento del `ListBox`.

```
Public Class Form11
    Private Sub Form11_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        Dim lst As New ListBox()
        lst.Items.Add("Rojo")
        lst.Items.Add("Verde")
        lst.Items.Add("Azul")
        lst.Location = New Point(50, 50)
        AddHandler lst.SelectedIndexChanged, AddressOf
Me.List_SelectedIndexChanged
        Me.Controls.Add(lst)
    End Sub

    Private Sub List_SelectedIndexChanged(sender As Object, e As EventArgs)
        Dim lst As ListBox = CType(sender, ListBox)
        MessageBox.Show("Seleccionaste: " & lst.SelectedItem)
    End Sub
End Class
```

Programa 12 – ComboBox básico

Enunciado: Muestra un menú desplegable con opciones.

```
Public Class Form12
    Private Sub Form12_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        Dim cmb As New ComboBox()
        cmb.Items.Add("Opción 1")
        cmb.Items.Add("Opción 2")
        cmb.Items.Add("Opción 3")
        cmb.Location = New Point(50, 50)
        Me.Controls.Add(cmb)
    End Sub
End Class
```

Programa 13 – ComboBox con evento

Enunciado: Muestra un mensaje cuando se selecciona una opción del `ComboBox`.

```
Public Class Form13
    Private Sub Form13_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        Dim cmb As New ComboBox()
        cmb.Items.Add("Perro")
        cmb.Items.Add("Gato")
        cmb.Items.Add("Pájaro")
        cmb.Location = New Point(50, 50)
        AddHandler cmb.SelectedIndexChanged, AddressOf
Me.Cmb_SelectedIndexChanged
        Me.Controls.Add(cmb)
    End Sub

    Private Sub Cmb_SelectedIndexChanged(sender As Object, e As EventArgs)
        Dim cmb As ComboBox = CType(sender, ComboBox)
        MessageBox.Show("Seleccionaste: " & cmb.SelectedItem)
    End Sub
End Class
```

Programa 14 – ListBox básico

Enunciado: Este programa permite al usuario seleccionar su fruta favorita de un ListBox y mostrar la selección en un Label.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        ' Agregar elementos al ListBox
        ListBox1.Items.Add("Manzana")
        ListBox1.Items.Add("Plátano")
        ListBox1.Items.Add("Naranja")
        ListBox1.Items.Add("Fresa")
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        ' Mostrar el elemento seleccionado
        If ListBox1.SelectedItem IsNot Nothing Then
            Label1.Text = "Fruta seleccionada: " &
ListBox1.SelectedItem.ToString()
        Else
            Label1.Text = "Seleccione una fruta"
        End If
    End Sub
End Class
```

Controles usados: ListBox, Button, Label.

Programa 15 – ComboBox básico

Enunciado: Permite al usuario elegir un color de un ComboBox y cambiar el color de fondo del formulario.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        ComboBox1.Items.Add("Rojo")
        ComboBox1.Items.Add("Verde")
        ComboBox1.Items.Add("Azul")
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Select Case ComboBox1.SelectedItem
            Case "Rojo"
                Me.BackColor = Color.Red
            Case "Verde"
                Me.BackColor = Color.Green
            Case "Azul"
                Me.BackColor = Color.Blue
            Case Else
                MessageBox.Show("Seleccione un color")
        End Select
    End Sub
End Class
```

Controles usados: ComboBox, Button.

Programa 16 – ListBox con múltiples selecciones

Enunciado: Permite seleccionar varias materias de un ListBox y mostrar todas las seleccionadas en un Label.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        ListBox1.SelectionMode = SelectionMode.MultiExtended
        ListBox1.Items.Add("Matemáticas")
        ListBox1.Items.Add("Física")
        ListBox1.Items.Add("Química")
        ListBox1.Items.Add("Historia")
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim materias As String = "Materias seleccionadas: "
        For Each item In ListBox1.SelectedItems
            materias &= item.ToString() & ", "
        Next
        Label1.Text = materias.TrimEnd(", ".ToCharArray())
    End Sub
End Class
```

Controles usados: ListBox, Button, Label.

Programa 17 – PictureBox

Enunciado: Muestra una imagen en un PictureBox al hacer clic en un botón.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        PictureBox1.Image = Image.FromFile("C:\Imagenes\foto.jpg") 'Ruta de la
imagen
        PictureBox1.SizeMode = PictureBoxSizeMode.StretchImage
    End Sub
End Class
```

Controles usados: PictureBox, Button.

Programa 18 – GroupBox con RadioButton

Enunciado: Permite elegir un método de pago dentro de un GroupBox y mostrar la selección en un Label.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        If RadioButton1.Checked Then
            Label1.Text = "Pago con Tarjeta"
        ElseIf RadioButton2.Checked Then
            Label1.Text = "Pago en Efectivo"
        ElseIf RadioButton3.Checked Then
            Label1.Text = "Pago con Transferencia"
        Else
            Label1.Text = "Seleccione un método de pago"
        End If
    End Sub
End Class
```

Controles usados: GroupBox, RadioButton, Button, Label.

Programa 19– Panel con botones dinámicos

Enunciado: Crea botones dentro de un Panel de forma dinámica y muestra cuál botón se pulsó.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        For i As Integer = 1 To 3
            Dim btn As New Button
            btn.Text = "Botón " & i
            btn.Top = 10 + (i - 1) * 40
            btn.Width = 100
            AddHandler btn.Click, AddressOf Boton_Click
            Panel1.Controls.Add(btn)
        Next
    End Sub

    Private Sub Boton_Click(sender As Object, e As EventArgs)
        Dim boton As Button = CType(sender, Button)
        MessageBox.Show("Presionaste " & boton.Text)
    End Sub
End Class
```

Controles usados: Panel, Button (dinámicos).

Ejercicios

Programa 1 – Cambiar el Text de un Label

Enunciado: Cambia el texto de un Label al hacer clic en un botón.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Label1.Text = "Hola, mundo!"
    End Sub
End Class
```

Programa 2 – Activar y desactivar un TextBox

Enunciado: Habilita o deshabilita un TextBox según el estado de un CheckBox.

```
Public Class Form1
    Private Sub CheckBox1_CheckedChanged(sender As Object, e As EventArgs)
Handles CheckBox1.CheckedChanged
        TextBox1.Enabled = CheckBox1.Checked
    End Sub
End Class
```

Programa 3 – Mostrar u ocultar un Button

Enunciado: Muestra u oculta un botón al hacer clic en otro botón.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Button2.Visible = Not Button2.Visible
    End Sub
End Class
```

Programa 4 – Cambiar fuente de un Label

Enunciado: Modifica la fuente y tamaño de un Label al hacer clic en un botón.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Label1.Font = New Font("Arial", 16, FontStyle.Bold)
    End Sub
End Class
```

Programa 5 – Cambiar color de fondo y texto

Enunciado: Cambia el BackColor y ForeColor de un TextBox.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        TextBox1.BackColor = Color.Yellow
        TextBox1.ForeColor = Color.Red
    End Sub
End Class
```

Programa 6 – Cambiar tamaño y ubicación de un botón

Enunciado: Ajusta el tamaño y la posición de un botón en tiempo de ejecución.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Button2.Size = New Size(150, 50)
        Button2.Location = New Point(100, 100)
    End Sub
End Class
```

Programa 7 – Ajustar Dock y Anchor

Enunciado: Modifica cómo un TextBox se ajusta al formulario usando Dock y Anchor.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        TextBox1.Dock = DockStyle.Top
        TextBox2.Anchor = AnchorStyles.Top Or AnchorStyles.Left Or
AnchorStyles.Right
    End Sub
End Class
```

Programa 8 – Cambiar título del formulario

Enunciado: Modifica la propiedad Text del formulario para mostrar un mensaje personalizado.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Me.Text = "Mi nuevo título"
    End Sub
End Class
```

Programa 9 – Cambiar color de fondo del formulario

Enunciado: Cambia el BackColor del formulario al hacer clic en un botón.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Me.BackColor = Color.LightBlue
    End Sub
End Class
```

Programa 10 – Enfocar un TextBox

Enunciado: Usa el método Focus para poner el cursor en un TextBox.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        TextBox1.Focus()
    End Sub
End Class
```

Programa 11 – Seleccionar todo el texto de un TextBox

Enunciado: Usa el método Select para seleccionar todo el contenido de un TextBox.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        TextBox1.SelectAll()
    End Sub
End Class
```

Programa 12 – Refrescar un Label

Enunciado: Actualiza la pantalla de un Label después de cambiar su texto usando Refresh.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Label1.Text = "Actualizando..."
        Label1.Refresh()
    End Sub
End Class
```

Programa 13 – Deshabilitar todos los botones

Enunciado: Recorre todos los controles del formulario y deshabilita los botones.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        For Each ctrl As Control In Me.Controls
            If TypeOf ctrl Is Button Then
                ctrl.Enabled = False
            End If
        Next
    End Sub
End Class
```

Programa 14 – Alternar visibilidad de Label y TextBox

Enunciado: Muestra u oculta Label y TextBox simultáneamente.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Label1.Visible = Not Label1.Visible
        TextBox1.Visible = Not TextBox1.Visible
    End Sub
End Class
```

Programa 15 – Cambiar fuente y color de un botón

Enunciado: Modifica la fuente y el color de un botón al pasar el mouse sobre él.

```
Public Class Form1
    Private Sub Button1_MouseHover(sender As Object, e As EventArgs) Handles
Button1.MouseHover
        Button1.Font = New Font("Verdana", 12, FontStyle.Italic)
        Button1.BackColor = Color.Orange
    End Sub
End Class
```

Programa 16 – Cambiar tamaño del formulario dinámicamente

Enunciado: Ajusta Width y Height del formulario con un botón.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Me.Size = New Size(600, 400)
    End Sub
End Class
```

Programa 17 – Cambiar posición del formulario

Enunciado: Mueve el formulario a una nueva ubicación en pantalla.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Me.Location = New Point(200, 150)
    End Sub
End Class
```

Programa 18 – Habilitar y deshabilitar controles por grupo

Enunciado: Activa o desactiva todos los TextBox de un Panel.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        For Each ctrl As Control In Panel1.Controls
            If TypeOf ctrl Is TextBox Then
                ctrl.Enabled = Not ctrl.Enabled
            End If
        Next
    End Sub
End Class
```

Programa 19 – Cambiar texto y color de varios controles

Enunciado: Cambia propiedades de varios controles simultáneamente.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Label1.Text = "Nuevo texto"
        Label1.ForeColor = Color.Green
        TextBox1.BackColor = Color.LightGray
        Button1.BackColor = Color.Yellow
    End Sub
End Class
```

Programa 20 – Refrescar formulario completo

Enunciado: Cambia el color del formulario y refresca todos los controles.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Me.BackColor = Color.LightPink
        Me.Refresh()
    End Sub
End Class
```

Ejercicios

Programa 1: Evento Click en un botón

Enunciado: Al hacer clic en un botón, se muestra un mensaje en un `MessageBox`.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        MessageBox.Show("Has hecho clic en el botón.")
    End Sub
End Class
```

Programa 2: Evento DoubleClick en un Label

Enunciado: Al hacer doble clic en un `Label`, su texto cambia.

```
Public Class Form1
    Private Sub Label1_DoubleClick(sender As Object, e As EventArgs) Handles
Label1.DoubleClick
        Label1.Text = "¡Has hecho doble clic!"
    End Sub
End Class
```

Programa 3: Evento MouseDown en un Panel

Enunciado: Detecta cuándo se presiona el botón del ratón sobre un `Panel`.

```
Public Class Form1
    Private Sub Panel1_MouseDown(sender As Object, e As MouseEventArgs) Handles
Panel1.MouseDown
        MessageBox.Show("Botón del ratón presionado en el panel.")
    End Sub
End Class
```

Programa 4: Evento MouseUp en un PictureBox

Enunciado: Muestra un mensaje cuando se suelta el botón del ratón sobre un `PictureBox`.

```
Public Class Form1
    Private Sub PictureBox1_MouseUp(sender As Object, e As MouseEventArgs)
Handles PictureBox1.MouseUp
        MessageBox.Show("Botón del ratón soltado en la imagen.")
    End Sub
End Class
```

Programa 5: Evento MouseMove en el Form

Enunciado: Cambia el título del formulario mostrando la posición del ratón.

```
Public Class Form1
    Private Sub Form1_MouseMove(sender As Object, e As MouseEventArgs) Handles
Me.MouseMove
        Me.Text = "X: " & e.X & " Y: " & e.Y
    End Sub
End Class
```

Programa 6: Evento KeyDown en un TextBox

Enunciado: Detecta la tecla presionada mientras se escribe en un `TextBox`.

```
Public Class Form1
    Private Sub TextBox1_KeyDown(sender As Object, e As KeyEventArgs) Handles
TextBox1.KeyDown
        Label1.Text = "Tecla presionada: " & e.KeyCode.ToString()
    End Sub
End Class
```

Programa 7: Evento KeyUp en un TextBox

Enunciado: Detecta cuándo se suelta una tecla y la muestra en un `Label`.

```
Public Class Form1
    Private Sub TextBox1_KeyUp(sender As Object, e As KeyEventArgs) Handles
TextBox1.KeyUp
        Label1.Text = "Tecla liberada: " & e.KeyCode.ToString()
    End Sub
End Class
```

Programa 8: Evento KeyPress en un TextBox

Enunciado: Detecta caracteres presionados y permite solo letras.

```
Public Class Form1
    Private Sub TextBox1_KeyPress(sender As Object, e As KeyPressEventArgs)
Handles TextBox1.KeyPress
        If Not Char.IsLetter(e.KeyChar) AndAlso Not Char.IsControl(e.KeyChar)
Then
            e.Handled = True
            MessageBox.Show("Solo se permiten letras.")
        End If
    End Sub
End Class
```

Programa 9: Evento Load del Form

Enunciado: Muestra un mensaje cuando el formulario se carga.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        MessageBox.Show("Formulario cargado.")
    End Sub
End Class
```

Programa 10: Evento Activated del Form

Enunciado: Cambia el color de fondo al activar el formulario.

```
Public Class Form1
    Private Sub Form1_Activated(sender As Object, e As EventArgs) Handles
Me.Activated
        Me.BackColor = Color.LightBlue
    End Sub
End Class
```

Programa 11: Evento FormClosing

Enunciado: Pregunta al usuario antes de cerrar el formulario.

```
Public Class Form1
    Private Sub Form1_FormClosing(sender As Object, e As FormClosingEventArgs)
Handles Me.FormClosing
        Dim result = MessageBox.Show("¿Deseas cerrar el formulario?", "Cerrar",
MessageBoxButtons.YesNo)
        If result = DialogResult.No Then e.Cancel = True
    End Sub
End Class
```

Programa 12: Evento FormClosed

Enunciado: Muestra un mensaje al cerrarse el formulario.

```
Public Class Form1
    Private Sub Form1_FormClosed(sender As Object, e As FormClosedEventArgs)
Handles Me.FormClosed
        MessageBox.Show("El formulario se ha cerrado.")
    End Sub
End Class
```

Programa 13: Evento Click y DoubleClick combinados

Enunciado: Diferencia entre clic simple y doble clic en un botón.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        MessageBox.Show("Clic simple.")
    End Sub
    Private Sub Button1_DoubleClick(sender As Object, e As EventArgs) Handles
Button1.DoubleClick
        MessageBox.Show("Doble clic.")
    End Sub
End Class
```

Programa 14: Evento MouseHover

Enunciado: Cambia el color de fondo de un botón cuando se pasa el ratón por encima.

```
Public Class Form1
    Private Sub Button1_MouseHover(sender As Object, e As EventArgs) Handles
Button1.MouseHover
        Button1.BackColor = Color.Yellow
    End Sub
End Class
```

Programa 15: Evento MouseLeave

Enunciado: Vuelve al color original cuando el ratón sale del botón.

```
Public Class Form1
    Private Sub Button1_MouseLeave(sender As Object, e As EventArgs) Handles
Button1.MouseLeave
        Button1.BackColor = SystemColors.Control
    End Sub
End Class
```

Programa 16: Crear un evento personalizado

Enunciado: Define un evento llamado `Saludo` y lo dispara al hacer clic en un botón.

```
Public Class Form1
    Public Event Saludo(sender As Object, e As EventArgs)
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        RaiseEvent Saludo(Me, EventArgs.Empty)
    End Sub
    Private Sub Form1_Saludo(sender As Object, e As EventArgs) Handles Me.Saludo
        MessageBox.Show("¡Hola desde un evento personalizado!")
    End Sub
End Class
```

Programa 17: Evento MouseClick en PictureBox

Enunciado: Detecta cuál botón del ratón se ha usado sobre una imagen.

```
Public Class Form1
    Private Sub PictureBox1_MouseClick(sender As Object, e As MouseEventArgs)
Handles PictureBox1.MouseClick
        If e.Button = MouseButtons.Left Then
            MessageBox.Show("Clic izquierdo en la imagen")
        ElseIf e.Button = MouseButtons.Right Then
            MessageBox.Show("Clic derecho en la imagen")
        End If
    End Sub
End Class
```

Programa 18: Evento MouseDoubleClick en ListBox

Enunciado: Muestra el elemento seleccionado al hacer doble clic.

```
Public Class Form1
    Private Sub ListBox1_DoubleClick(sender As Object, e As EventArgs) Handles
ListBox1.DoubleClick
        If ListBox1.SelectedItem IsNot Nothing Then
            MessageBox.Show("Has seleccionado: " &
ListBox1.SelectedItem.ToString())
        End If
    End Sub
End Class
```

Programa 19: Evento KeyDown con combinación de teclas

Enunciado: Detecta si se presiona Ctrl + S y muestra un mensaje.

```
Public Class Form1
    Private Sub Form1_KeyDown(sender As Object, e As KeyEventArgs) Handles
Me.KeyDown
        If e.Control AndAlso e.KeyCode = Keys.S Then
            MessageBox.Show("Has presionado Ctrl+S")
        End If
    End Sub
End Class
```

Programa 20: Evento TextChanged en TextBox

Enunciado: Actualiza un `Label` en tiempo real conforme se escribe en un `TextBox`.

```
Public Class Form1
    Private Sub TextBox1_TextChanged(sender As Object, e As EventArgs) Handles
TextBox1.TextChanged
        Label1.Text = "Texto actual: " & TextBox1.Text
    End Sub
End Class
```

Parte IV – Controles Avanzados

12. Menús y Barras de Herramientas

Los menús y barras de herramientas permiten ofrecer opciones de manera organizada al usuario, facilitando la interacción con la aplicación. En VB.NET, los controles principales son **MenuStrip**, **ToolStrip** y **ContextMenuStrip**.

12.1. MenuStrip: creación de menús y submenús

MenuStrip permite crear menús desplegables en la parte superior del formulario.

Ejemplo de creación de un menú:

1. Arrastrar un **MenuStrip** desde la caja de herramientas al formulario.
2. Agregar elementos de menú, por ejemplo: **Archivo → Nuevo, Abrir, Guardar** y **Editar → Copiar, Pegar**.

Código para manejar un clic en un elemento de menú:

```
Private Sub NuevoToolStripMenuItem_Click(sender As Object, e As EventArgs)
Handles NuevoToolStripMenuItem.Click
    MessageBox.Show("Se seleccionó 'Nuevo'")
End Sub

Private Sub AbrirToolStripMenuItem_Click(sender As Object, e As EventArgs)
Handles AbrirToolStripMenuItem.Click
    MessageBox.Show("Se seleccionó 'Abrir'")
End Sub
```

12.2. ToolStrip: botones e iconos en barras de herramientas

ToolStrip permite agregar botones, cuadros de texto, menús desplegables e iconos.

Ejemplo de barra de herramientas con botones:

1. Arrastrar un **ToolStrip** al formulario.
2. Agregar botones con iconos representativos, como **Nuevo**, **Abrir**, **Guardar**.

Código de manejo de botones:

```
Private Sub tsbNuevo_Click(sender As Object, e As EventArgs) Handles
tsbNuevo.Click
    MessageBox.Show("Botón Nuevo presionado")
End Sub

Private Sub tsbGuardar_Click(sender As Object, e As EventArgs) Handles
tsbGuardar.Click
    MessageBox.Show("Botón Guardar presionado")
End Sub
```

12.3. ContextMenuStrip: menús contextuales

ContextMenuStrip permite mostrar menús al hacer clic derecho sobre un control.

Ejemplo: menú contextual sobre un TextBox:

1. Arrastrar un **ContextMenuStrip** al formulario y agregar elementos, por ejemplo: **Cortar, Copiar, Pegar**.
2. Asignarlo al TextBox: `TextBox1.ContextMenuStrip = ContextMenuStrip1`

Código de manejo:

```
Private Sub CopiarToolStripMenuItem_Click(sender As Object, e As EventArgs)
Handles CopiarToolStripMenuItem.Click
    TextBox1.Copy()
End Sub

Private Sub PegarToolStripMenuItem_Click(sender As Object, e As EventArgs)
Handles PegarToolStripMenuItem.Click
    TextBox1.Paste()
End Sub
```

12.4. Menús dinámicos en tiempo de ejecución

Se pueden crear menús de manera **dinámica** mediante código, sin necesidad de agregarlos desde el diseñador.

Ejemplo de menú dinámico:

```
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    Dim menu As New MenuStrip()
    Me.Controls.Add(menu)

    ' Crear menú Archivo
    Dim archivo As New ToolStripMenuItem("Archivo")
    menu.Items.Add(archivo)

    ' Crear submenú Nuevo
    Dim nuevo As New ToolStripMenuItem("Nuevo")
    AddHandler nuevo.Click, AddressOf Nuevo_Click
    archivo.DropDownItems.Add(nuevo)

    ' Crear submenú Salir
    Dim salir As New ToolStripMenuItem("Salir")
    AddHandler salir.Click, AddressOf Salir_Click
    archivo.DropDownItems.Add(salir)
End Sub

Private Sub Nuevo_Click(sender As Object, e As EventArgs)
    MessageBox.Show("Nuevo elemento creado dinámicamente")
End Sub

Private Sub Salir_Click(sender As Object, e As EventArgs)
    Me.Close()
End Sub
```

---

13. Controles de Organización y Navegación

Estos controles permiten organizar la información dentro del formulario y facilitar la navegación dentro de la aplicación. Los principales son **TabControl**, **SplitContainer**, **TreeView** y **ListView**.

13.1. TabControl

TabControl permite organizar la interfaz en **pestañas**, cada una con diferentes controles y contenido.

Ejemplo de TabControl:

1. Arrastrar un **TabControl** al formulario.
2. Agregar **TabPages**: `TabPage1`, `TabPage2`, `TabPage3`.
3. Colocar controles en cada pestaña según sea necesario.

Código para manejar el cambio de pestaña:

```
Private Sub TabControl1_SelectedIndexChanged(sender As Object, e As EventArgs)
Handles TabControl1.SelectedIndexChanged

    MessageBox.Show("Pestaña actual: " & TabControl1.SelectedTab.Text)

End Sub
```

13.2. SplitContainer

SplitContainer permite dividir un formulario en **paneles redimensionables**, útiles para organizar contenido de manera flexible.

Ejemplo de SplitContainer:

1. Arrastrar un **SplitContainer** al formulario.
2. Configurar orientación: `Orientation = Horizontal` o `Vertical`.
3. Colocar controles en **Panel1** y **Panel2**.

Código para manejar el cambio de tamaño de los paneles:

```
Private Sub SplitContainer1_SplitterMoved(sender As Object, e As
SplitterEventArgs) Handles SplitContainer1.SplitterMoved

    Label1.Text = "Panel1 ancho: " & SplitContainer1.Panel1.Width & ", Panel2
ancho: " & SplitContainer1.Panel2.Width

End Sub
```

13.3. TreeView

TreeView permite mostrar información jerárquica, como carpetas y archivos, en un **árbol expandible**.

Ejemplo de TreeView:

1. Arrastrar un **TreeView** al formulario.
2. Agregar nodos en diseño o mediante código.

```
TreeView1.Nodes.Add("Raíz")

TreeView1.Nodes(0).Nodes.Add("Hijo 1")

TreeView1.Nodes(0).Nodes.Add("Hijo 2")

TreeView1.Nodes(0).Nodes(1).Nodes.Add("Nieto 1")
```

Manejo de evento de selección:

```
Private Sub TreeView1_AfterSelect(sender As Object, e As TreeViewEventArgs)
Handles TreeView1.AfterSelect

    MessageBox.Show("Nodo seleccionado: " & e.Node.Text)

End Sub
```

13.4. ListView

ListView permite mostrar datos en formato **lista, detalles, iconos grandes o pequeños**, ideal para aplicaciones de inventario o exploradores de archivos.

Ejemplo de ListView en modo detalles:

1. Arrastrar un **ListView** al formulario.
2. Configurar `View = Details` y agregar columnas:

```
ListView1.Columns.Add("Nombre", 150)
ListView1.Columns.Add("Edad", 50)
ListView1.Columns.Add("Ciudad", 100)
```

3. Agregar elementos:

```
Dim item As New ListViewItem("Ana")
item.SubItems.Add("25")
item.SubItems.Add("Madrid")
ListView1.Items.Add(item)
```

Manejo de evento de selección:

```
Private Sub ListView1_SelectedIndexChanged(sender As Object, e As EventArgs)
Handles ListView1.SelectedIndexChanged
    If ListView1.SelectedItems.Count > 0 Then
        MessageBox.Show("Elemento seleccionado: " &
ListView1.SelectedItems(0).Text)
    End If
End Sub
```

---

14. Controles de Datos

Los **controles de datos** permiten mostrar, organizar y manipular información proveniente de bases de datos u otras fuentes de datos. Los principales controles son **DataGridView** y los mecanismos de **Data Binding**.

14.1. DataGridView

DataGridView es un control que permite mostrar datos en forma de tabla, con filas y columnas, y facilita la edición, ordenamiento y selección de registros.

Ejemplo de uso básico:

1. Arrastrar un **DataGridView** al formulario.
2. Agregar columnas desde el diseñador o mediante código:

```
DataGridView1.ColumnCount = 3

DataGridView1.Columns(0).Name = "ID"

DataGridView1.Columns(1).Name = "Nombre"

DataGridView1.Columns(2).Name = "Edad"
```

3. Agregar filas:

```
Dim row As String() = {"1", "Ana", "25"}

DataGridView1.Rows.Add(row)


row = {"2", "Luis", "30"}

DataGridView1.Rows.Add(row)
```

Manejo de evento de selección:

```
Private Sub DataGridView1_SelectionChanged(sender As Object, e As EventArgs)
Handles DataGridView1.SelectionChanged

    If DataGridView1.SelectedRows.Count > 0 Then

        MessageBox.Show("Fila seleccionada: " &
DataGridView1.SelectedRows(0).Cells(1).Value.ToString())

    End If

End Sub
```

14.2. Enlaces de datos (Data Binding)

El **Data Binding** permite conectar controles directamente a **fuentes de datos** como listas, objetos o bases de datos, sin necesidad de manipular los datos manualmente.

Ejemplo con lista de objetos:

```
Public Class Persona
    Public Property ID As Integer
    Public Property Nombre As String
    Public Property Edad As Integer
End Class

Dim lista As New List(Of Persona) From {
    New Persona With {.ID = 1, .Nombre = "Ana", .Edad = 25},
    New Persona With {.ID = 2, .Nombre = "Luis", .Edad = 30}
}

Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    DataGridView1.DataSource = lista
End Sub
```

14.3. Cargar y mostrar datos en tablas

DataGridView puede vincularse a **DataTables**, lo que facilita mostrar datos de bases de datos o de otras fuentes en tiempo de ejecución.

Ejemplo con DataTable:

```
Dim tabla As New DataTable
tabla.Columns.Add("ID", GetType(Integer))
tabla.Columns.Add("Nombre", GetType(String))
tabla.Columns.Add("Edad", GetType(Integer))

tabla.Rows.Add(1, "Ana", 25)
tabla.Rows.Add(2, "Luis", 30)

DataGridView1.DataSource = tabla
```

Actualizar datos en tiempo de ejecución:

```
tabla.Rows(0)("Nombre") = "Ana María"
DataGridView1.Refresh()
```

Ejercicios

BLOQUE 1 – MenuStrip (Menús y submenús)

Programa 1 – Menú simple con dos opciones

Enunciado: Crear un formulario con un `MenuStrip` que tenga un menú llamado "Archivo" y dos subopciones: "Abrir" y "Salir". Al hacer clic en "Salir", el programa se cierra.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim menu As New MenuStrip()
        Dim archivo As New ToolStripMenuItem("Archivo")
        Dim abrir As New ToolStripMenuItem("Abrir")
        Dim salir As New ToolStripMenuItem("Salir")
        AddHandler salir.Click, Sub() Me.Close()
        archivo.DropDownItems.Add(abrir)
        archivo.DropDownItems.Add(salir)
        menu.Items.Add(archivo)
        Me.Controls.Add(menu)
        Me.MainMenuStrip = menu
    End Sub
End Class
```

Programa 2 – Menú con varias secciones

Enunciado: Crear un `MenuStrip` con tres menús: "Archivo", "Editar" y "Ayuda". Cada menú tiene al menos dos subopciones.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim menu As New MenuStrip()
        Dim archivo As New ToolStripMenuItem("Archivo")
        Dim abrir As New ToolStripMenuItem("Abrir")
        Dim salir As New ToolStripMenuItem("Salir")
        AddHandler salir.Click, Sub() Me.Close()
        archivo.DropDownItems.AddRange(New ToolStripItem() {abrir, salir})

        Dim editar As New ToolStripMenuItem("Editar")
        editar.DropDownItems.Add("Copiar")
        editar.DropDownItems.Add("Pegar")

        Dim ayuda As New ToolStripMenuItem("Ayuda")
        ayuda.DropDownItems.Add("Acerca de")

        menu.Items.AddRange(New ToolStripItem() {archivo, editar, ayuda})
        Me.Controls.Add(menu)
        Me.MainMenuStrip = menu
    End Sub
End Class
```

Programa 3 – Menú con atajos de teclado
Enunciado: Crear un menú "Archivo" con subopciones "Nuevo" y "Cerrar", asignando atajos Ctrl+N y Ctrl+X respectivamente.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim menu As New MenuStrip()
        Dim archivo As New ToolStripMenuItem("Archivo")
        Dim nuevo As New ToolStripMenuItem("Nuevo")
        nuevo.ShortcutKeys = Keys.Control Or Keys.N
        Dim cerrar As New ToolStripMenuItem("Cerrar")
        cerrar.ShortcutKeys = Keys.Control Or Keys.X
        AddHandler cerrar.Click, Sub() Me.Close()
        archivo.DropDownItems.AddRange(New ToolStripItem() {nuevo, cerrar})
        menu.Items.Add(archivo)
        Me.Controls.Add(menu)
        Me.MainMenuStrip = menu
    End Sub
End Class
```

Programa 4 – Menú con iconos
Enunciado: Crear un `MenuStrip` donde cada submenú tenga un icono asociado.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim menu As New MenuStrip()
        Dim archivo As New ToolStripMenuItem("Archivo")
        Dim abrir As New ToolStripMenuItem("Abrir")
        abrir.Image = My.Resources.abrir_icon 'Reemplazar con tu recurso de
imagen
        Dim salir As New ToolStripMenuItem("Salir")
        salir.Image = My.Resources.salir_icon
        AddHandler salir.Click, Sub() Me.Close()
        archivo.DropDownItems.AddRange(New ToolStripItem() {abrir, salir})
        menu.Items.Add(archivo)
        Me.Controls.Add(menu)
        Me.MainMenuStrip = menu
    End Sub
End Class
```

Programa 5 – Menú con items activables/desactivables
Enunciado: Crear un menú donde "Guardar" se desactive inicialmente y se active al hacer clic en "Nuevo".

```
Public Class Form1
    Private guardarItem As ToolStripMenuItem
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim menu As New MenuStrip()
        Dim archivo As New ToolStripMenuItem("Archivo")
        Dim nuevo As New ToolStripMenuItem("Nuevo")
        guardarItem = New ToolStripMenuItem("Guardar")
        guardarItem.Enabled = False
        AddHandler nuevo.Click, Sub() guardarItem.Enabled = True
        archivo.DropDownItems.AddRange(New ToolStripItem() {nuevo, guardarItem})
        menu.Items.Add(archivo)
        Me.Controls.Add(menu)
        Me.MainMenuStrip = menu
    End Sub
End Class
```

BLOQUE 2 – ToolStrip (Botones e iconos en barras de herramientas)

Programa 6 – Barra de herramientas simple
Enunciado: Crear un `ToolStrip` con botones "Nuevo", "Abrir" y "Salir".

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim tool As New ToolStrip()
        Dim btnNuevo As New ToolStripButton("Nuevo")
        Dim btnAbrir As New ToolStripButton("Abrir")
        Dim btnSalir As New ToolStripButton("Salir")
        AddHandler btnSalir.Click, Sub() Me.Close()
        tool.Items.AddRange(New ToolStripItem() {btnNuevo, btnAbrir, btnSalir})
        Me.Controls.Add(tool)
    End Sub
End Class
```

Programa 7 – Barra con iconos
Enunciado: Crear un `ToolStrip` donde cada botón tenga un icono.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim tool As New ToolStrip()
        Dim btnNuevo As New ToolStripButton()
        btnNuevo.Image = My.Resources.nuevo_icon
        Dim btnAbrir As New ToolStripButton()
        btnAbrir.Image = My.Resources.abrir_icon
        Dim btnSalir As New ToolStripButton()
        btnSalir.Image = My.Resources.salir_icon
        AddHandler btnSalir.Click, Sub() Me.Close()
        tool.Items.AddRange(New ToolStripItem() {btnNuevo, btnAbrir, btnSalir})
        Me.Controls.Add(tool)
    End Sub
End Class
```

Programa 8 – Botones con eventos
Enunciado: Cada botón de la barra muestra un mensaje diferente al ser clicado.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim tool As New ToolStrip()
        Dim btn1 As New ToolStripButton("Botón 1")
        Dim btn2 As New ToolStripButton("Botón 2")
        AddHandler btn1.Click, Sub() MessageBox.Show("Has pulsado Botón 1")
        AddHandler btn2.Click, Sub() MessageBox.Show("Has pulsado Botón 2")
        tool.Items.AddRange(New ToolStripItem() {btn1, btn2})
        Me.Controls.Add(tool)
    End Sub
End Class
```

Programa 9 – Botones con ToolTip
Enunciado: Agregar tooltips a los botones del `ToolStrip`.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim tool As New ToolStrip()
        Dim btn As New ToolStripButton("Info")
        btn.ToolTipText = "Haz clic para ver información"
        AddHandler btn.Click, Sub() MessageBox.Show("Información mostrada")
        tool.Items.Add(btn)
        Me.Controls.Add(tool)
    End Sub
End Class
```

Programa 10 – Separadores en ToolStrip
Enunciado: Crear una barra con botones separados por `ToolStripSeparator`.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim tool As New ToolStrip()
        Dim btn1 As New ToolStripButton("Uno")
        Dim btn2 As New ToolStripButton("Dos")
        Dim sep As New ToolStripSeparator()
        Dim btn3 As New ToolStripButton("Tres")
        tool.Items.AddRange(New ToolStripItem() {btn1, btn2, sep, btn3})
        Me.Controls.Add(tool)
    End Sub
End Class
```

---

11. Menú principal simple (MenuStrip)

Enunciado: Crea un formulario con un menú principal que tenga los elementos: Archivo, Editar y Ayuda.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim menu As New MenuStrip()
```

```
        Dim archivo As New ToolStripMenuItem("Archivo")
        archivo.DropDownItems.Add("Nuevo")
        archivo.DropDownItems.Add("Abrir")
        archivo.DropDownItems.Add("Salir")

        Dim editar As New ToolStripMenuItem("Editar")
        editar.DropDownItems.Add("Cortar")
        editar.DropDownItems.Add("Copiar")
        editar.DropDownItems.Add("Pegar")

        Dim ayuda As New ToolStripMenuItem("Ayuda")
        ayuda.DropDownItems.Add("Acerca de")

        menu.Items.AddRange(New ToolStripItem() {archivo, editar, ayuda})
        Me.MainMenuStrip = menu
        Me.Controls.Add(menu)
    End Sub
End Class
```

---

12. Menú con submenús y eventos

Enunciado: Crear un menú con submenús y mostrar un mensaje al hacer clic.

```
Public Class Form2
    Private Sub Form2_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim menu As New MenuStrip()

        Dim archivo As New ToolStripMenuItem("Archivo")
        Dim abrir As New ToolStripMenuItem("Abrir")
        AddHandler abrir.Click, Sub() MessageBox.Show("Abrir archivo
seleccionado")
        archivo.DropDownItems.Add(abrir)

        Dim salir As New ToolStripMenuItem("Salir")
        AddHandler salir.Click, Sub() Me.Close()
        archivo.DropDownItems.Add(salir)

        menu.Items.Add(archivo)
        Me.MainMenuStrip = menu
        Me.Controls.Add(menu)
    End Sub
End Class
```

---

13. Barra de herramientas con botones (ToolStrip)

Enunciado: Crear una barra de herramientas con botones de Nuevo, Abrir y Guardar.

```
Public Class Form3
    Private Sub Form3_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim toolbar As New ToolStrip()

        Dim btnNuevo As New ToolStripButton("Nuevo")
        AddHandler btnNuevo.Click, Sub() MessageBox.Show("Nuevo documento")
        Dim btnAbrir As New ToolStripButton("Abrir")
        AddHandler btnAbrir.Click, Sub() MessageBox.Show("Abrir archivo")
        Dim btnGuardar As New ToolStripButton("Guardar")
        AddHandler btnGuardar.Click, Sub() MessageBox.Show("Guardar archivo")

        toolbar.Items.AddRange(New ToolStripItem() {btnNuevo, btnAbrir,
btnGuardar})
        Me.Controls.Add(toolbar)
    End Sub
End Class
```

---

14. Menú contextual (ContextMenuStrip)

Enunciado: Crear un menú que aparezca al hacer clic derecho sobre el formulario.

```
Public Class Form4

    Private Sub Form4_Load(sender As Object, e As EventArgs) Handles MyBase.Load

        Dim menuContexto As New ContextMenuStrip()

        menuContexto.Items.Add("Copiar", Nothing, Sub()
MessageBox.Show("Copiado"))

        menuContexto.Items.Add("Pegar", Nothing, Sub()
MessageBox.Show("Pegado"))


        Me.ContextMenuStrip = menuContexto

    End Sub

End Class
```

---

15. Menú dinámico en tiempo de ejecución

Enunciado: Crear un menú que agregue un ítem nuevo al hacer clic en un botón.

```
Public Class Form5

    Private menu As New MenuStrip()

    Private Sub Form5_Load(sender As Object, e As EventArgs) Handles MyBase.Load

        Dim archivo As New ToolStripMenuItem("Archivo")

        menu.Items.Add(archivo)

        Me.MainMenuStrip = menu

        Me.Controls.Add(menu)


        Dim btnAgregar As New Button()

        btnAgregar.Text = "Agregar Item"

        btnAgregar.Top = 50

        AddHandler btnAgregar.Click, Sub()

                                        archivo.DropDownItems.Add("Nuevo Item")

                                    End Sub

        Me.Controls.Add(btnAgregar)

    End Sub

End Class
```

---

16. MenuStrip con atajos de teclado

Enunciado: Crear un menú con accesos rápidos (Ctrl+N, Ctrl+O).

```
Public Class Form6
    Private Sub Form6_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim menu As New MenuStrip()
        Dim archivo As New ToolStripMenuItem("Archivo")
        Dim nuevo As New ToolStripMenuItem("Nuevo")
        nuevo.ShortcutKeys = Keys.Control Or Keys.N
        AddHandler nuevo.Click, Sub() MessageBox.Show("Nuevo archivo")
        archivo.DropDownItems.Add(nuevo)

        Dim abrir As New ToolStripMenuItem("Abrir")
        abrir.ShortcutKeys = Keys.Control Or Keys.O
        AddHandler abrir.Click, Sub() MessageBox.Show("Abrir archivo")
        archivo.DropDownItems.Add(abrir)

        menu.Items.Add(archivo)
        Me.MainMenuStrip = menu
        Me.Controls.Add(menu)
    End Sub
End Class
```

---

17. ToolStrip con iconos

Enunciado: Crear una barra de herramientas con botones que tengan iconos.

```
Public Class Form7
    Private Sub Form7_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim toolbar As New ToolStrip()
        Dim btnNuevo As New ToolStripButton()
        btnNuevo.Text = "Nuevo"
        btnNuevo.Image = My.Resources.nuevo 'Reemplazar con tu icono
        AddHandler btnNuevo.Click, Sub() MessageBox.Show("Nuevo documento")
        toolbar.Items.Add(btnNuevo)

        Me.Controls.Add(toolbar)
    End Sub
End Class
```

18. Menú contextual en un TextBox

Enunciado: Mostrar menú contextual solo en un TextBox.

```
Public Class Form8
    Private Sub Form8_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim txt As New TextBox()
        txt.Top = 50
        txt.Width = 200
        Me.Controls.Add(txt)

        Dim menu As New ContextMenuStrip()
        menu.Items.Add("Cortar", Nothing, Sub() txt.Cut())
        menu.Items.Add("Copiar", Nothing, Sub() txt.Copy())
        menu.Items.Add("Pegar", Nothing, Sub() txt.Paste())

        txt.ContextMenuStrip = menu
    End Sub
End Class
```

---

19. Menú dinámico con submenús

Enunciado: Agregar submenús dinámicamente al hacer clic en un botón.

```
Public Class Form9
    Private menu As New MenuStrip()
    Private Sub Form9_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim archivo As New ToolStripMenuItem("Archivo")
        menu.Items.Add(archivo)
        Me.MainMenuStrip = menu
        Me.Controls.Add(menu)

        Dim btnAgregar As New Button()
        btnAgregar.Text = "Agregar Submenú"
        btnAgregar.Top = 50
        AddHandler btnAgregar.Click, Sub()
                                        archivo.DropDownItems.Add("Submenú " &
archivo.DropDownItems.Count + 1)
                                    End Sub
        Me.Controls.Add(btnAgregar)
    End Sub
End Class
```

---

20. ToolStrip con DropDownButton

Enunciado: Crear una ToolStrip que tenga un botón con menú desplegable.

```
Public Class Form10
    Private Sub Form10_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        Dim toolbar As New ToolStrip()
        Dim dropBtn As New ToolStripDropDownButton("Opciones")
        dropBtn.DropDownItems.Add("Opción 1", Nothing, Sub()
MessageBox.Show("Opción 1"))
        dropBtn.DropDownItems.Add("Opción 2", Nothing, Sub()
MessageBox.Show("Opción 2"))
        toolbar.Items.Add(dropBtn)
        Me.Controls.Add(toolbar)
    End Sub
End Class
```

---

21. MenuStrip con CheckBoxes

Enunciado: Crear un menú donde algunos ítems tengan casillas seleccionables.

```
Public Class Form11

    Private Sub Form11_Load(sender As Object, e As EventArgs) Handles
MyBase.Load

        Dim menu As New MenuStrip()

        Dim ver As New ToolStripMenuItem("Ver")

        Dim mostrarLineas As New ToolStripMenuItem("Mostrar líneas")

        mostrarLineas.CheckOnClick = True

        ver.DropDownItems.Add(mostrarLineas)

        menu.Items.Add(ver)

        Me.MainMenuStrip = menu

        Me.Controls.Add(menu)

    End Sub

End Class
```

---

22. ContextMenuStrip con imágenes

Enunciado: Menú contextual que muestra iconos junto a los ítems.

```
Public Class Form12

    Private Sub Form12_Load(sender As Object, e As EventArgs) Handles
MyBase.Load

        Dim menu As New ContextMenuStrip()

        menu.Items.Add(New ToolStripMenuItem("Cortar",
My.Resources.cortar_icono, Sub() MessageBox.Show("Cortar")))

        menu.Items.Add(New ToolStripMenuItem("Copiar",
My.Resources.copiar_icono, Sub() MessageBox.Show("Copiar")))

        Me.ContextMenuStrip = menu

    End Sub

End Class
```

Ejercicios

---

1. TabControl básico

Enunciado: Crear un TabControl con 2 pestañas y etiquetas en cada una.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim tabs As New TabControl()
        tabs.Dock = DockStyle.Fill

        Dim tab1 As New TabPage("Pestaña 1")
        tab1.Controls.Add(New Label With {.Text = "Contenido de la Pestaña
1", .Dock = DockStyle.Fill})
        Dim tab2 As New TabPage("Pestaña 2")
        tab2.Controls.Add(New Label With {.Text = "Contenido de la Pestaña
2", .Dock = DockStyle.Fill})

        tabs.TabPages.Add(tab1)
        tabs.TabPages.Add(tab2)

        Me.Controls.Add(tabs)
    End Sub
End Class
```

---

2. TabControl con botones

Enunciado: Cada pestaña contiene un botón que muestra un mensaje.

```
Public Class Form2
    Private Sub Form2_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim tabs As New TabControl() With {.Dock = DockStyle.Fill}

        Dim tab1 As New TabPage("Inicio")
        Dim btn1 As New Button() With {.Text = "Click aquí"}
        AddHandler btn1.Click, Sub() MessageBox.Show("Botón de la pestaña
Inicio")
        tab1.Controls.Add(btn1)

        Dim tab2 As New TabPage("Opciones")
        Dim btn2 As New Button() With {.Text = "Click aquí"}
        AddHandler btn2.Click, Sub() MessageBox.Show("Botón de la pestaña
Opciones")
        tab2.Controls.Add(btn2)

        tabs.TabPages.Add(tab1)
        tabs.TabPages.Add(tab2)
        Me.Controls.Add(tabs)
    End Sub
End Class
```

---

3. TabControl con pestañas dinámicas

Enunciado: Agregar nuevas pestañas mediante un botón.

```
Public Class Form3
    Private tabControl As New TabControl() With {.Dock = DockStyle.Fill}
    Private Sub Form3_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Me.Controls.Add(tabControl)
        Dim btn As New Button() With {.Text = "Agregar Pestaña", .Top = 200}
        AddHandler btn.Click, Sub()
                                    Dim tab As New TabPage("Pestaña " &
tabControl.TabCount + 1)
                                    tab.Controls.Add(New Label With {.Text =
"Nueva pestaña"})
                                    tabControl.TabPages.Add(tab)
                                End Sub
        Me.Controls.Add(btn)
    End Sub
End Class
```

---

4. SplitContainer horizontal

Enunciado: Dividir el formulario horizontalmente en dos paneles con controles distintos.

```
Public Class Form4
    Private Sub Form4_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim split As New SplitContainer() With {.Dock =
DockStyle.Fill, .Orientation = Orientation.Horizontal}
        split.Panel1.Controls.Add(New Label With {.Text = "Panel
Superior", .Dock = DockStyle.Fill})
        split.Panel2.Controls.Add(New Label With {.Text = "Panel
Inferior", .Dock = DockStyle.Fill})
        Me.Controls.Add(split)
    End Sub
End Class
```

---

5. SplitContainer vertical con botones

Enunciado: Dividir el formulario verticalmente y colocar botones en cada panel.

```
Public Class Form5
    Private Sub Form5_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim split As New SplitContainer() With {.Dock =
DockStyle.Fill, .Orientation = Orientation.Vertical}

        Dim btn1 As New Button() With {.Text = "Botón Izquierdo", .Dock =
DockStyle.Fill}
        Dim btn2 As New Button() With {.Text = "Botón Derecho", .Dock =
DockStyle.Fill}

        split.Panel1.Controls.Add(btn1)
        split.Panel2.Controls.Add(btn2)

        Me.Controls.Add(split)
    End Sub
End Class
```

---

6. TreeView básico

Enunciado: Crear un TreeView con nodos principales y subnodos.

```
Public Class Form6
    Private Sub Form6_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim tree As New TreeView() With {.Dock = DockStyle.Fill}
        Dim nodo1 As New TreeNode("Frutas")
        nodo1.Nodes.Add("Manzana")
        nodo1.Nodes.Add("Banana")
        Dim nodo2 As New TreeNode("Verduras")
        nodo2.Nodes.Add("Zanahoria")
        nodo2.Nodes.Add("Lechuga")
        tree.Nodes.Add(nodo1)
        tree.Nodes.Add(nodo2)
        Me.Controls.Add(tree)
    End Sub
End Class
```

---

7. TreeView con evento de selección

Enunciado: Mostrar mensaje al seleccionar un nodo del TreeView.

```
Public Class Form7
    Private Sub Form7_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim tree As New TreeView() With {.Dock = DockStyle.Fill}
        tree.Nodes.Add("Animales").Nodes.Add("Perro")
        tree.Nodes(0).Nodes.Add("Gato")
        AddHandler tree.AfterSelect, Sub(sender2, e2)
                                            MessageBox.Show("Seleccionaste: " &
e2.Node.Text)
                                        End Sub
        Me.Controls.Add(tree)
    End Sub
End Class
```

---

8. TreeView con nodos dinámicos

Enunciado: Agregar nodos al TreeView mediante un botón.

```
Public Class Form8
    Private tree As New TreeView() With {.Dock = DockStyle.Left, .Width = 200}
    Private Sub Form8_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Me.Controls.Add(tree)
        Dim btn As New Button() With {.Text = "Agregar Nodo", .Top = 220}
        AddHandler btn.Click, Sub()
                                    Dim nodo As New TreeNode("Nodo " &
tree.Nodes.Count + 1)
                                    tree.Nodes.Add(nodo)
                                End Sub
        Me.Controls.Add(btn)
    End Sub
End Class
```

---

9. ListView en modo detalles

Enunciado: Mostrar una lista con columnas y datos.

```
Public Class Form9
    Private Sub Form9_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim lv As New ListView() With {.Dock = DockStyle.Fill, .View =
View.Details}
        lv.Columns.Add("Nombre", 100)
        lv.Columns.Add("Edad", 50)
        lv.Items.Add(New ListViewItem(New String() {"Ana", "25"}))
        lv.Items.Add(New ListViewItem(New String() {"Luis", "30"}))
        Me.Controls.Add(lv)
    End Sub
End Class
```

---

10. ListView con iconos

Enunciado: Mostrar imágenes junto a los elementos de la lista.

```
Public Class Form10
    Private Sub Form10_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        Dim lv As New ListView() With {.Dock = DockStyle.Fill, .View =
View.LargeIcon}
        Dim imgs As New ImageList()
        imgs.Images.Add(My.Resources.icono1) 'Agregar recurso de icono
        imgs.Images.Add(My.Resources.icono2)
        lv.LargeImageList = imgs

        lv.Items.Add(New ListViewItem("Elemento 1", 0))
        lv.Items.Add(New ListViewItem("Elemento 2", 1))
        Me.Controls.Add(lv)
    End Sub
End Class
```

---

11. TabControl con TreeView en cada pestaña

Enunciado: Cada pestaña del TabControl contiene un TreeView con nodos distintos.

```
Public Class Form11

    Private Sub Form11_Load(sender As Object, e As EventArgs) Handles
MyBase.Load

        Dim tabs As New TabControl() With {.Dock = DockStyle.Fill}


        Dim tab1 As New TabPage("Animales")

        Dim tree1 As New TreeView() With {.Dock = DockStyle.Fill}

        tree1.Nodes.Add("Mamíferos").Nodes.Add("Perro")

        tree1.Nodes(0).Nodes.Add("Gato")

        tab1.Controls.Add(tree1)


        Dim tab2 As New TabPage("Frutas")

        Dim tree2 As New TreeView() With {.Dock = DockStyle.Fill}

        tree2.Nodes.Add("Cítricos").Nodes.Add("Naranja")

        tree2.Nodes(0).Nodes.Add("Limón")

        tab2.Controls.Add(tree2)


        tabs.TabPages.Add(tab1)

        tabs.TabPages.Add(tab2)

        Me.Controls.Add(tabs)

    End Sub

End Class
```

---

12. SplitContainer con ListView y botones

Enunciado: Panel izquierdo con ListView, panel derecho con botones para manipular la lista.

```
Public Class Form12

    Private Sub Form12_Load(sender As Object, e As EventArgs) Handles
MyBase.Load

        Dim split As New SplitContainer() With {.Dock =
DockStyle.Fill, .Orientation = Orientation.Vertical}

        Dim lv As New ListView() With {.Dock = DockStyle.Fill, .View =
View.Details}

        lv.Columns.Add("Nombre", 100)

        lv.Columns.Add("Edad", 50)

        lv.Items.Add(New ListViewItem(New String() {"Ana", "25"}))

        lv.Items.Add(New ListViewItem(New String() {"Luis", "30"}))

        split.Panel1.Controls.Add(lv)


        Dim btnAdd As New Button() With {.Text = "Agregar Persona", .Top = 10}

        AddHandler btnAdd.Click, Sub()

                                        lv.Items.Add(New ListViewItem(New String()
{"Nuevo", "0"}))

                                    End Sub

        split.Panel2.Controls.Add(btnAdd)


        Me.Controls.Add(split)

    End Sub

End Class
```

---

13. TabControl + SplitContainer

Enunciado: Una pestaña con SplitContainer, otra con ListView.

```
Public Class Form13

    Private Sub Form13_Load(sender As Object, e As EventArgs) Handles
MyBase.Load

        Dim tabs As New TabControl() With {.Dock = DockStyle.Fill}


        ' Pestaña 1: SplitContainer

        Dim tab1 As New TabPage("Paneles")

        Dim split As New SplitContainer() With {.Dock = DockStyle.Fill}

        split.Panel1.Controls.Add(New Label() With {.Text = "Panel 1"})

        split.Panel2.Controls.Add(New Label() With {.Text = "Panel 2"})

        tab1.Controls.Add(split)


        ' Pestaña 2: ListView

        Dim tab2 As New TabPage("Lista")

        Dim lv As New ListView() With {.Dock = DockStyle.Fill, .View =
View.Details}

        lv.Columns.Add("Elemento", 150)

        lv.Items.Add("Item 1")

        lv.Items.Add("Item 2")

        tab2.Controls.Add(lv)


        tabs.TabPages.Add(tab1)

        tabs.TabPages.Add(tab2)

        Me.Controls.Add(tabs)

    End Sub

End Class
```

---

14. TreeView con ListView sincronizado

Enunciado: Seleccionar un nodo en TreeView muestra detalles en ListView.

```
Public Class Form14

    Private Sub Form14_Load(sender As Object, e As EventArgs) Handles
MyBase.Load

        Dim split As New SplitContainer() With {.Dock =
DockStyle.Fill, .Orientation = Orientation.Vertical}

        Dim tree As New TreeView() With {.Dock = DockStyle.Fill}

        tree.Nodes.Add("Frutas").Nodes.Add("Manzana")

        tree.Nodes(0).Nodes.Add("Banana")

        split.Panel1.Controls.Add(tree)


        Dim lv As New ListView() With {.Dock = DockStyle.Fill, .View =
View.Details}

        lv.Columns.Add("Detalle", 200)

        split.Panel2.Controls.Add(lv)


        AddHandler tree.AfterSelect, Sub(sender2, e2)

                                            lv.Items.Clear()

                                            lv.Items.Add("Seleccionaste: " &
e2.Node.Text)

                                        End Sub


        Me.Controls.Add(split)

    End Sub

End Class
```

---

15. TabControl dinámico con SplitContainer

Enunciado: Agregar nuevas pestañas con SplitContainer mediante un botón.

```
Public Class Form15

    Private tabs As New TabControl() With {.Dock = DockStyle.Fill}

    Private Sub Form15_Load(sender As Object, e As EventArgs) Handles
MyBase.Load

        Me.Controls.Add(tabs)

        Dim btn As New Button() With {.Text = "Agregar Pestaña", .Top = 200}

        AddHandler btn.Click, Sub()

                                    Dim tab As New TabPage("Pestaña " &
tabs.TabCount + 1)

                                    Dim split As New SplitContainer() With {.Dock
= DockStyle.Fill}

                                    split.Panel1.Controls.Add(New Label() With
{.Text = "Panel 1"})

                                    split.Panel2.Controls.Add(New Label() With
{.Text = "Panel 2"})

                                    tab.Controls.Add(split)

                                    tabs.TabPages.Add(tab)

                                End Sub

        Me.Controls.Add(btn)

    End Sub

End Class
```

---

16. TreeView con nodos y ListView por nodo

Enunciado: Cada nodo muestra diferentes elementos en ListView.

```
Public Class Form16

    Private Sub Form16_Load(sender As Object, e As EventArgs) Handles
MyBase.Load

        Dim split As New SplitContainer() With {.Dock = DockStyle.Fill}

        Dim tree As New TreeView() With {.Dock = DockStyle.Fill}

        tree.Nodes.Add("Animales")

        tree.Nodes.Add("Frutas")

        split.Panel1.Controls.Add(tree)


        Dim lv As New ListView() With {.Dock = DockStyle.Fill, .View =
View.Details}

        lv.Columns.Add("Elemento", 200)

        split.Panel2.Controls.Add(lv)


        AddHandler tree.AfterSelect, Sub(sender2, e2)

                                        lv.Items.Clear()

                                        Select Case e2.Node.Text

                                            Case "Animales"

                                                lv.Items.Add("Perro")

                                                lv.Items.Add("Gato")

                                            Case "Frutas"

                                                lv.Items.Add("Manzana")

                                                lv.Items.Add("Banana")

                                        End Select

                                    End Sub


        Me.Controls.Add(split)

    End Sub

End Class
```

---

17. ListView con selección múltiple y detalles

Enunciado: Seleccionar varios elementos en ListView muestra un mensaje con todos ellos.

```
Public Class Form17

    Private Sub Form17_Load(sender As Object, e As EventArgs) Handles
MyBase.Load

        Dim lv As New ListView() With {.Dock = DockStyle.Fill, .View =
View.Details, .MultiSelect = True}

        lv.Columns.Add("Nombre", 150)

        lv.Items.Add("Ana")

        lv.Items.Add("Luis")

        lv.Items.Add("Pedro")

        AddHandler lv.SelectedIndexChanged, Sub()

                                                    If lv.SelectedItems.Count > 0
Then

                                                        Dim nombres As String =
String.Join(", ", lv.SelectedItems.Cast(Of ListViewItem).Select(Function(i)
i.Text))


MessageBox.Show("Seleccionaste: " & nombres)

                                                    End If

                                                End Sub

        Me.Controls.Add(lv)

    End Sub

End Class
```

---

18. TabControl + TreeView + ListView integrado

Enunciado: Una pestaña contiene TreeView, otra ListView; interacción entre ellos.

```
Public Class Form18

    Private Sub Form18_Load(sender As Object, e As EventArgs) Handles
MyBase.Load

        Dim tabs As New TabControl() With {.Dock = DockStyle.Fill}


        Dim tab1 As New TabPage("Árbol")

        Dim tree As New TreeView() With {.Dock = DockStyle.Fill}

        tree.Nodes.Add("Colores").Nodes.Add("Rojo")

        tree.Nodes(0).Nodes.Add("Azul")

        tab1.Controls.Add(tree)


        Dim tab2 As New TabPage("Lista")

        Dim lv As New ListView() With {.Dock = DockStyle.Fill, .View =
View.Details}

        lv.Columns.Add("Elemento", 150)

        tab2.Controls.Add(lv)


        AddHandler tree.AfterSelect, Sub(sender2, e2)

                                            lv.Items.Clear()

                                            lv.Items.Add("Nodo seleccionado: " &
e2.Node.Text)

                                        End Sub


        tabs.TabPages.Add(tab1)

        tabs.TabPages.Add(tab2)

        Me.Controls.Add(tabs)

    End Sub

End Class
```

Ejercicios

PROGRAMA 1

Enunciado: Mostrar una tabla simple en un `DataGridView` con datos generados manualmente en el código.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        ' Crear DataTable
        Dim dt As New DataTable
        dt.Columns.Add("ID")
        dt.Columns.Add("Nombre")
        dt.Columns.Add("Edad")

        ' Agregar filas
        dt.Rows.Add("1", "Ana", "25")
        dt.Rows.Add("2", "Luis", "30")
        dt.Rows.Add("3", "María", "22")

        ' Asignar al DataGridView
        DataGridView1.DataSource = dt

        ' Mostrar el código en un TextBox
        TextBox1.Text = Me.GetType().ToString()
    End Sub
End Class
```

PROGRAMA 2

Enunciado: Mostrar cómo enlazar un `List(Of Object)` a un `DataGridView`.

```
Public Class Form1
    Class Persona
        Public Property ID As Integer
        Public Property Nombre As String
        Public Property Edad As Integer
    End Class

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim lista As New List(Of Persona) From {
            New Persona With {.ID = 1, .Nombre = "Carlos", .Edad = 28},
            New Persona With {.ID = 2, .Nombre = "Lucía", .Edad = 35},
            New Persona With {.ID = 3, .Nombre = "Pedro", .Edad = 40}
        }

        ' Enlace de datos
        DataGridView1.DataSource = lista
        TextBox1.Text = "Ejemplo de Data Binding con List(Of Object)"
    End Sub
End Class
```

PROGRAMA 3

Enunciado: Cargar datos desde un `DataTable` y permitir la edición directa en el `DataGridView`.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim dt As New DataTable
        dt.Columns.Add("Producto")
        dt.Columns.Add("Precio")

        dt.Rows.Add("Laptop", 1200)
        dt.Rows.Add("Mouse", 25)
        dt.Rows.Add("Teclado", 45)

        DataGridView1.DataSource = dt
        DataGridView1.AllowUserToAddRows = True
        DataGridView1.AllowUserToDeleteRows = True

        TextBox1.Text = "Edición directa en DataGridView"
    End Sub
End Class
```

PROGRAMA 4

Enunciado: Uso de `BindingSource` para manejar el enlace de datos entre `DataTable` y `DataGridView`.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim dt As New DataTable
        dt.Columns.Add("Código")
        dt.Columns.Add("Descripción")

        dt.Rows.Add("A01", "Producto A")
        dt.Rows.Add("B02", "Producto B")

        Dim bs As New BindingSource()
        bs.DataSource = dt

        DataGridView1.DataSource = bs
        TextBox1.Text = "DataGridView usando BindingSource"
    End Sub
End Class
```

PROGRAMA 5

Enunciado: Mostrar cómo enlazar un `ComboBox` a un `DataGridView` usando Data Binding.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim dt As New DataTable
        dt.Columns.Add("ID")
        dt.Columns.Add("Ciudad")

        dt.Rows.Add(1, "Madrid")
        dt.Rows.Add(2, "Barcelona")
        dt.Rows.Add(3, "Sevilla")

        DataGridView1.DataSource = dt
        ComboBox1.DataSource = dt
        ComboBox1.DisplayMember = "Ciudad"
        ComboBox1.ValueMember = "ID"

        TextBox1.Text = "Data Binding con ComboBox y DataGridView"
    End Sub
End Class
```

PROGRAMA 6

Enunciado: Mostrar una tabla simple en un `DataGridView` con datos generados manualmente.

```
Public Class Form1

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

        Dim dt As New DataTable

        dt.Columns.Add("ID")

        dt.Columns.Add("Nombre")

        dt.Columns.Add("Edad")


        dt.Rows.Add("1", "Ana", "25")

        dt.Rows.Add("2", "Luis", "30")

        dt.Rows.Add("3", "María", "22")


        DataGridView1.DataSource = dt

        TextBox1.Text = "Mostrar tabla simple en DataGridView"

    End Sub

End Class
```

PROGRAMA 7

Enunciado: Enlazar un `List(Of Object)` a un `DataGridView`.

```
Public Class Form1
    Class Persona
        Public Property ID As Integer
        Public Property Nombre As String
        Public Property Edad As Integer
    End Class

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim lista As New List(Of Persona) From {
            New Persona With {.ID = 1, .Nombre = "Carlos", .Edad = 28},
            New Persona With {.ID = 2, .Nombre = "Lucía", .Edad = 35},
            New Persona With {.ID = 3, .Nombre = "Pedro", .Edad = 40}
        }

        DataGridView1.DataSource = lista
        TextBox1.Text = "Enlace de datos con List(Of Object)"
    End Sub
End Class
```

PROGRAMA 8

Enunciado: Permitir edición directa en un `DataGridView` cargado desde un `DataTable`.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim dt As New DataTable
        dt.Columns.Add("Producto")
        dt.Columns.Add("Precio")

        dt.Rows.Add("Laptop", 1200)
        dt.Rows.Add("Mouse", 25)
        dt.Rows.Add("Teclado", 45)

        DataGridView1.DataSource = dt
        DataGridView1.AllowUserToAddRows = True
        DataGridView1.AllowUserToDeleteRows = True

        TextBox1.Text = "Edición directa en DataGridView"
    End Sub
End Class
```

PROGRAMA 9

Enunciado: Uso de `BindingSource` para enlazar `DataTable` a `DataGridView`.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim dt As New DataTable
        dt.Columns.Add("Código")
        dt.Columns.Add("Descripción")

        dt.Rows.Add("A01", "Producto A")
        dt.Rows.Add("B02", "Producto B")

        Dim bs As New BindingSource()
        bs.DataSource = dt

        DataGridView1.DataSource = bs
        TextBox1.Text = "DataGridView usando BindingSource"
    End Sub
End Class
```

PROGRAMA 10

Enunciado: Enlazar un `ComboBox` a un `DataGridView` usando Data Binding.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim dt As New DataTable
        dt.Columns.Add("ID")
        dt.Columns.Add("Ciudad")

        dt.Rows.Add(1, "Madrid")
        dt.Rows.Add(2, "Barcelona")
        dt.Rows.Add(3, "Sevilla")

        DataGridView1.DataSource = dt
        ComboBox1.DataSource = dt
        ComboBox1.DisplayMember = "Ciudad"
        ComboBox1.ValueMember = "ID"

        TextBox1.Text = "Data Binding con ComboBox y DataGridView"
    End Sub
End Class
```

PROGRAMA 11

Enunciado: Ordenar columnas del `DataGridView` al hacer clic en el encabezado.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim dt As New DataTable
        dt.Columns.Add("ID")
        dt.Columns.Add("Nombre")

        dt.Rows.Add(3, "Juan")
        dt.Rows.Add(1, "Ana")
        dt.Rows.Add(2, "Luis")

        DataGridView1.DataSource = dt
        DataGridView1.Sort(DataGridView1.Columns("ID"),
System.ComponentModel.ListSortDirection.Ascending)

        TextBox1.Text = "Ordenar columnas al hacer clic"
    End Sub
End Class
```

PROGRAMA 12

Enunciado: Filtrar datos del `DataGridView` usando `BindingSource`.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim dt As New DataTable
        dt.Columns.Add("Nombre")
        dt.Columns.Add("Ciudad")

        dt.Rows.Add("Ana", "Madrid")
        dt.Rows.Add("Luis", "Barcelona")
        dt.Rows.Add("María", "Madrid")

        Dim bs As New BindingSource()
        bs.DataSource = dt
        DataGridView1.DataSource = bs

        ' Filtrar solo Madrid
        bs.Filter = "Ciudad='Madrid'"

        TextBox1.Text = "Filtrar DataGridView por ciudad"
    End Sub
End Class
```

PROGRAMA 13

Enunciado: Mostrar datos en `DataGridView` desde un archivo CSV.

```
Imports System.IO

Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim dt As New DataTable
        dt.Columns.Add("ID")
        dt.Columns.Add("Nombre")
        dt.Columns.Add("Edad")

        For Each line As String In File.ReadAllLines("datos.csv")
            Dim parts() As String = line.Split(","c)
            dt.Rows.Add(parts)
        Next

        DataGridView1.DataSource = dt
        TextBox1.Text = "Cargar datos desde CSV"
    End Sub
End Class
```

PROGRAMA 14

Enunciado: Mostrar la fila seleccionada en un `Label`.

```
Public Class Form1

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

        Dim dt As New DataTable

        dt.Columns.Add("ID")

        dt.Columns.Add("Nombre")

        dt.Rows.Add(1, "Ana")

        dt.Rows.Add(2, "Luis")

        DataGridView1.DataSource = dt


        TextBox1.Text = "Mostrar fila seleccionada en Label"

    End Sub


    Private Sub DataGridView1_SelectionChanged(sender As Object, e As EventArgs)
Handles DataGridView1.SelectionChanged

        If DataGridView1.CurrentRow IsNot Nothing Then

            Label1.Text =
DataGridView1.CurrentRow.Cells("Nombre").Value.ToString()

        End If

    End Sub

End Class
```

PROGRAMA 15

Enunciado: Agregar una nueva fila al `DataGridView` mediante un botón.

```
Public Class Form1
    Dim dt As New DataTable

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        dt.Columns.Add("ID")
        dt.Columns.Add("Nombre")
        DataGridView1.DataSource = dt
        TextBox1.Text = "Agregar fila con botón"
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        dt.Rows.Add(TextBoxID.Text, TextBoxNombre.Text)
    End Sub
End Class
```

PROGRAMA 16

Enunciado: Eliminar la fila seleccionada del `DataGridView` mediante un botón.

```
Public Class Form1
    Dim dt As New DataTable

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        dt.Columns.Add("ID")
        dt.Columns.Add("Nombre")
        dt.Rows.Add(1, "Ana")
        dt.Rows.Add(2, "Luis")
        DataGridView1.DataSource = dt
        TextBox1.Text = "Eliminar fila seleccionada"
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        If DataGridView1.CurrentRow IsNot Nothing Then
            DataGridView1.Rows.Remove(DataGridView1.CurrentRow)
        End If
    End Sub
End Class
```

PROGRAMA 17

Enunciado: Actualizar una celda del `DataGridView` mediante un botón.

```
Public Class Form1
    Dim dt As New DataTable

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        dt.Columns.Add("ID")
        dt.Columns.Add("Nombre")
        dt.Rows.Add(1, "Ana")
        dt.Rows.Add(2, "Luis")
        DataGridView1.DataSource = dt
        TextBox1.Text = "Actualizar celda seleccionada"
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        If DataGridView1.CurrentRow IsNot Nothing Then
            DataGridView1.CurrentRow.Cells("Nombre").Value = TextBoxNombre.Text
        End If
    End Sub
End Class
```

PROGRAMA 18

Enunciado: Mostrar la suma de valores de una columna numérica.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim dt As New DataTable
        dt.Columns.Add("Producto")
        dt.Columns.Add("Precio", GetType(Double))
        dt.Rows.Add("Laptop", 1200)
        dt.Rows.Add("Mouse", 25)
        dt.Rows.Add("Teclado", 45)

        DataGridView1.DataSource = dt

        Dim suma As Double = dt.AsEnumerable().Sum(Function(r) r.Field(Of
Double)("Precio"))
        Label1.Text = "Suma de precios: " & suma
        TextBox1.Text = "Suma de columna numérica"
    End Sub
End Class
```

PROGRAMA 19

Enunciado: Mostrar datos en `DataGridView` desde un `Array`.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim personas() As String = {"Ana,25", "Luis,30", "María,22"}
        Dim dt As New DataTable
        dt.Columns.Add("Nombre")
        dt.Columns.Add("Edad")

        For Each p In personas
            Dim parts() As String = p.Split(","c)
            dt.Rows.Add(parts)
        Next

        DataGridView1.DataSource = dt
        TextBox1.Text = "Cargar DataGridView desde Array"
    End Sub
End Class
```

PROGRAMA 20

Enunciado: Resaltar la fila seleccionada en el `DataGridView`.

```
Public Class Form1

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

        Dim dt As New DataTable

        dt.Columns.Add("ID")

        dt.Columns.Add("Nombre")

        dt.Rows.Add(1, "Ana")

        dt.Rows.Add(2, "Luis")

        DataGridView1.DataSource = dt

        TextBox1.Text = "Resaltar fila seleccionada"

    End Sub


    Private Sub DataGridView1_SelectionChanged(sender As Object, e As EventArgs)
Handles DataGridView1.SelectionChanged

        For Each row As DataGridViewRow In DataGridView1.Rows

            row.DefaultCellStyle.BackColor = If(row.Selected, Color.Yellow,
Color.White)

        Next

    End Sub

End Class
```

Parte V – Funcionalidades Específicas

15. Funciones y Procedimientos

En **VB.NET**, los **procedimientos** y **funciones** permiten organizar el código en bloques reutilizables, mejorar la legibilidad y evitar duplicación de código.

15.1. Diferencia entre `Function` y `Sub`

- **`Sub`**: procedimiento que realiza acciones pero **no devuelve un valor**.
- **`Function`**: procedimiento que **devuelve un valor** al final de su ejecución.

Ejemplo de Sub:

```
Sub MostrarMensaje(ByVal texto As String)
    MessageBox.Show(texto)
End Sub
```

Ejemplo de Function:

```
Function Sumar(a As Integer, b As Integer) As Integer
    Return a + b
End Function
```

Uso:

```
MostrarMensaje("Hola Mundo")         ' Llama a la Sub
Dim resultado As Integer
resultado = Sumar(5, 3)              ' Llama a la Function
MessageBox.Show("Suma: " & resultado)
```

15.2. Paso de parámetros por valor y por referencia

- **Por valor (`ByVal`)**: se pasa **una copia** del valor, los cambios dentro de la función **no afectan la variable original**.
- **Por referencia (`ByRef`)**: se pasa **la referencia** a la variable, los cambios dentro de la función **modifican la variable original**.

Ejemplo:

```
Sub IncrementarValor(ByVal x As Integer)
    x += 1
End Sub

Sub IncrementarReferencia(ByRef x As Integer)
    x += 1
End Sub

Dim num As Integer = 5
IncrementarValor(num)      ' num sigue siendo 5
IncrementarReferencia(num) ' num ahora es 6
```

15.3. Funciones integradas de VB.NET

VB.NET incluye **funciones predefinidas** para operaciones comunes:

- **Matemáticas:** `Math.Abs()`, `Math.Sqrt()`, `Math.Pow()`, `Math.Round()`
- **Cadenas:** `Len()`, `Left()`, `Right()`, `Mid()`, `Trim()`, `Replace()`
- **Conversión de tipos:** `CInt()`, `CDbl()`, `CStr()`
- **Fecha y hora:** `Now()`, `DateAdd()`, `DateDiff()`, `Format()`

Ejemplo:

```
Dim texto As String = "  Hola Mundo  "
Dim longitud As Integer = Len(texto)          ' 13
Dim limpio As String = Trim(texto)            ' "Hola Mundo"

Dim raiz As Double = Math.Sqrt(25)            ' 5
Dim potencia As Double = Math.Pow(2, 3)       ' 8
```

15.4. Funciones recursivas

Una **función recursiva** es aquella que se llama a sí misma para resolver un problema más grande dividiéndolo en subproblemas más pequeños.

Ejemplo: Factorial de un número

```
Function Factorial(n As Integer) As Integer
    If n <= 1 Then
        Return 1
    Else
        Return n * Factorial(n - 1)
    End If
End Function

MessageBox.Show("Factorial de 5: " & Factorial(5)) ' Resultado: 120
```

Notas sobre recursión:

- Debe existir una **condición de parada** para evitar bucles infinitos.
- Ideal para problemas que se pueden dividir en subproblemas idénticos (factoriales, Fibonacci, búsquedas en estructuras jerárquicas).

---

16. Mensajes y Diálogos

En Windows Forms, mostrar mensajes y permitir la interacción con el usuario mediante cuadros de diálogo es fundamental para crear aplicaciones intuitivas. VB.NET ofrece diversas herramientas para esto, como `MsgBox`, `MessageBox` y varios cuadros de diálogo predefinidos.

16.1. Uso de `MsgBox`

`MsgBox` es la forma más sencilla de mostrar un mensaje al usuario.

Sintaxis básica:

```
MsgBox("Texto del mensaje")
```

Ejemplo con título y estilo:

```
MsgBox("Operación completada", MsgBoxStyle.Information, "Aviso")
```

Estilos más comunes:

- `MsgBoxStyle.OkOnly` – Solo botón OK
- `MsgBoxStyle.Critical` – Icono de error
- `MsgBoxStyle.Exclamation` – Icono de advertencia
- `MsgBoxStyle.Information` – Icono de información
- `MsgBoxStyle.YesNo` – Botones Sí y No

Ejemplo con decisión del usuario:

```
Dim respuesta As MsgBoxResult

respuesta = MsgBox("¿Desea continuar?", MsgBoxStyle.YesNo +
MsgBoxStyle.Question, "Confirmar")


If respuesta = MsgBoxResult.Yes Then

    MsgBox("Continuando...")

Else

    MsgBox("Operación cancelada")

End If
```

16.2. `MessageBox` con botones y estilos

`MessageBox` es más moderno y flexible que `MsgBox`. Permite más control sobre botones, iconos y opciones.

Sintaxis básica:

```
MessageBox.Show("Texto del mensaje")
```

Ejemplo con título, botones e icono:

```
Dim resultado As DialogResult

resultado = MessageBox.Show("¿Desea guardar los cambios?", "Guardar archivo",
MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question)


If resultado = DialogResult.Yes Then

    MessageBox.Show("Guardando...")

ElseIf resultado = DialogResult.No Then

    MessageBox.Show("No se guardó")

Else

    MessageBox.Show("Cancelado")

End If
```

Opciones de botones (`MessageBoxButtons`):

- `OK`
- `OKCancel`
- `YesNo`
- `YesNoCancel`
- `RetryCancel`

Opciones de iconos (`MessageBoxIcon`):

- `Information`
- `Exclamation`
- `Error`
- `Question`

16.3. Cuadros de diálogo comunes

VB.NET incluye cuadros de diálogo integrados para tareas habituales: abrir, guardar, seleccionar color o fuente.

16.3.1. `OpenFileDialog`

Permite al usuario **seleccionar un archivo existente**.

```
Dim abrir As New OpenFileDialog
abrir.Title = "Seleccionar archivo"
abrir.Filter = "Archivos de texto|*.txt|Todos los archivos|*.*"

If abrir.ShowDialog() = DialogResult.OK Then
    Dim ruta As String = abrir.FileName
    MessageBox.Show("Archivo seleccionado: " & ruta)
End If
```

16.3.2. `SaveFileDialog`

Permite al usuario **guardar un archivo**.

```
Dim guardar As New SaveFileDialog
guardar.Title = "Guardar archivo"
guardar.Filter = "Archivos de texto|*.txt"

If guardar.ShowDialog() = DialogResult.OK Then
    Dim ruta As String = guardar.FileName
    MessageBox.Show("Archivo guardado en: " & ruta)
End If
```

16.3.3. `ColorDialog`

Permite seleccionar un color.

```
Dim colorDlg As New ColorDialog

If colorDlg.ShowDialog() = DialogResult.OK Then
    Me.BackColor = colorDlg.Color
End If
```

16.3.4. `FontDialog`

Permite seleccionar una fuente y estilo de texto.

```
Dim fuenteDlg As New FontDialog

If fuenteDlg.ShowDialog() = DialogResult.OK Then
    Label1.Font = fuenteDlg.Font
End If
```

---

17. Manejo de Texto y Formato

En VB.NET, trabajar con texto y formatos de datos es fundamental para procesar información de formularios, mostrar resultados al usuario y validar entradas. Este capítulo abarca técnicas para manipular cadenas, dar formato a números y texto, convertir tipos de datos y realizar validaciones básicas.

17.1. Manipulación de cadenas (`String`, `StringBuilder`)

Las cadenas (`String`) son inmutables en VB.NET, lo que significa que cada operación genera una nueva cadena. Para modificaciones intensivas, se recomienda `StringBuilder`.

Operaciones básicas con `String`:

```
Dim texto As String = "Hola Mundo"
Dim longitud As Integer = texto.Length                ' Longitud de la cadena
Dim mayus As String = texto.ToUpper()                ' Convertir a mayúsculas
Dim minus As String = texto.ToLower()                ' Convertir a minúsculas
Dim subCadena As String = texto.Substring(0, 4)    ' Extraer "Hola"
Dim reemplazo As String = texto.Replace("Mundo", "VB.NET")  ' Reemplazo
Dim contiene As Boolean = texto.Contains("Mundo")   ' Verifica existencia
```

Uso de `StringBuilder` para eficiencia:

```
Imports System.Text

Dim sb As New StringBuilder("Hola")
sb.Append(" Mundo")
sb.Insert(0, "Mensaje: ")
sb.Replace("Hola", "Hola a todos")
Dim resultado As String = sb.ToString()
MessageBox.Show(resultado)
```

17.2. Formato de cadenas y números

VB.NET permite dar formato a cadenas y números con `String.Format` o expresiones de interpolación (`$""`).

Ejemplos de formato numérico:

```
Dim valor As Double = 1234.567

MessageBox.Show(String.Format("Número con 2 decimales: {0:N2}", valor))   '
1,234.57

MessageBox.Show(String.Format("Moneda: {0:C}", valor))                     '
$1,234.57

MessageBox.Show(String.Format("Porcentaje: {0:P}", 0.25))                  '
25.00%
```

Ejemplo de interpolación:

```
Dim nombre As String = "Ana"

Dim edad As Integer = 25

MessageBox.Show($"Nombre: {nombre}, Edad: {edad}")
```

17.3. Funciones de conversión de datos

Para trabajar con diferentes tipos de datos, VB.NET ofrece funciones de conversión:

- `CInt(valor)` → Convierte a entero
- `CDbl(valor)` → Convierte a doble
- `CStr(valor)` → Convierte a cadena
- `CDate(valor)` → Convierte a fecha
- `CBool(valor)` → Convierte a booleano

Ejemplo práctico:

```
Dim textoNumero As String = "123"

Dim numero As Integer = CInt(textoNumero)          ' Convierte a entero

Dim textoConvertido As String = CStr(numero)       ' Convierte de nuevo a cadena

Dim precio As Double = CDbl("45.67")               ' Convierte a doble
```

Manejo de errores al convertir:

```
Dim valor As String = "abc"
Dim numero As Integer
If Integer.TryParse(valor, numero) Then
    MessageBox.Show("Número válido: " & numero)
Else
    MessageBox.Show("Error: el valor no es un número")
End If
```

17.4. Funciones de validación

Validar entradas es clave para aplicaciones robustas. Algunos métodos comunes:

Validación de texto vacío:

```
If String.IsNullOrWhiteSpace(TextBox1.Text) Then
    MessageBox.Show("Debe ingresar un valor")
End If
```

Validación de formato de correo electrónico:

```
Imports System.Text.RegularExpressions

Dim email As String = TextBoxEmail.Text
Dim patron As String = "^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$"
If Regex.IsMatch(email, patron) Then
    MessageBox.Show("Correo válido")
Else
    MessageBox.Show("Correo inválido")
End If
```

Validación de números en un rango:

```
Dim numero As Integer
If Integer.TryParse(TextBoxNumero.Text, numero) Then
    If numero >= 1 And numero <= 100 Then
        MessageBox.Show("Número dentro del rango")
    Else
        MessageBox.Show("Número fuera del rango")
    End If
Else
    MessageBox.Show("No es un número válido")
End If
```

18. Manejo de Fechas y Horas

En VB.NET, trabajar con fechas y horas es fundamental para aplicaciones que manejan calendarios, horarios, registros y planificaciones. Este capítulo explica cómo declarar, manipular, comparar y formatear fechas y horas utilizando la estructura `Date` y la clase `DateTime`.

18.1. Declaración de variables `Date`

En VB.NET, las fechas se manejan con el tipo `Date` o la clase `DateTime`.

Declaración de variables:

```
Dim fechaActual As Date = Date.Now              ' Fecha y hora actual
Dim fechaNacimiento As Date = #1990-05-15#     ' Fecha específica
Dim horaActual As Date = TimeOfDay              ' Solo la hora actual
```

Notas:

- Los literales de fecha se encierran entre #.
- `Date.Now` devuelve fecha y hora actuales.
- `TimeOfDay` devuelve solo la hora actual.

18.2. Propiedades y métodos de `DateTime`

`DateTime` ofrece propiedades y métodos útiles:

Propiedades comunes:

```
Dim fecha As Date = Date.Now
MessageBox.Show(fecha.Year)       ' Año actual
MessageBox.Show(fecha.Month)      ' Mes actual
MessageBox.Show(fecha.Day)        ' Día actual
MessageBox.Show(fecha.Hour)       ' Hora actual
MessageBox.Show(fecha.Minute)     ' Minutos actuales
MessageBox.Show(fecha.Second)     ' Segundos actuales
MessageBox.Show(fecha.DayOfWeek)  ' Día de la semana
MessageBox.Show(fecha.DayOfYear)  ' Día del año
```

Métodos útiles:

```
Dim fecha As Date = Date.Now
Dim fechaMasUnaSemana As Date = fecha.AddDays(7)   ' Sumar días
Dim fechaMesSiguiente As Date = fecha.AddMonths(1) ' Sumar meses
Dim fechaAñoAnterior As Date = fecha.AddYears(-1)  ' Restar años
```

18.3. Operaciones con fechas (sumar, restar, comparar)

Suma y resta de fechas:

```
Dim hoy As Date = Date.Now
Dim mañana As Date = hoy.AddDays(1)
Dim ayer As Date = hoy.AddDays(-1)
```

Diferencia entre fechas:

```
Dim fechaInicio As Date = #2025-09-01#
Dim fechaFin As Date = #2025-09-27#
Dim diferencia As TimeSpan = fechaFin - fechaInicio
MessageBox.Show("Días entre fechas: " & diferencia.Days)
```

Comparación de fechas:

```
Dim fecha1 As Date = #2025-09-27#
Dim fecha2 As Date = #2025-10-01#

If fecha1 < fecha2 Then
    MessageBox.Show("fecha1 es anterior a fecha2")
ElseIf fecha1 > fecha2 Then
    MessageBox.Show("fecha1 es posterior a fecha2")
Else
    MessageBox.Show("Las fechas son iguales")
End If
```

18.4. Formato de fecha y hora

VB.NET permite mostrar fechas y horas en distintos formatos mediante `ToString` o `String.Format`.

Formatos comunes:

```
Dim fecha As Date = Date.Now

MessageBox.Show(fecha.ToString("dd/MM/yyyy"))       ' 27/09/2025
MessageBox.Show(fecha.ToString("MM-dd-yyyy"))       ' 09-27-2025
MessageBox.Show(fecha.ToString("yyyy-MM-dd HH:mm")) ' 2025-09-27 14:30
MessageBox.Show(fecha.ToString("dddd, dd MMMM yyyy")) ' sábado, 27 septiembre
2025
```

Interpolación con formato:

```
MessageBox.Show($"Fecha actual: {fecha:dd/MM/yyyy} Hora: {fecha:HH:mm:ss}")
```

Ejercicios

PROGRAMA 1: Función simple (Function)

Enunciado: Crear un formulario que calcule el cuadrado de un número usando `Function`.

Explicación: `Function` devuelve un valor.

```
Public Class Form1
    Private Function Cuadrado(ByVal num As Double) As Double
        Return num * num
    End Function

    Private Sub btnCalcular_Click(sender As Object, e As EventArgs) Handles
btnCalcular.Click
        Dim numero As Double = CDbl(txtNumero.Text)
        lblResultado.Text = "Cuadrado: " & Cuadrado(numero)
    End Sub
End Class
```

PROGRAMA 2: Procedimiento simple (Sub)

Enunciado: Mostrar un mensaje en un `Label` usando `Sub`.

Explicación: `Sub` ejecuta código pero **no devuelve valor**.

```
Public Class Form1
    Private Sub MostrarMensaje(ByVal mensaje As String)
        lblMensaje.Text = mensaje
    End Sub

    Private Sub btnMostrar_Click(sender As Object, e As EventArgs) Handles
btnMostrar.Click
        MostrarMensaje("Hola, este es un procedimiento Sub")
    End Sub
End Class
```

PROGRAMA 3: Diferencia entre Function y Sub

Enunciado: Comparar un `Function` que devuelve un valor y un `Sub` que solo muestra mensaje.

```
Public Class Form1
    Private Function Sumar(a As Double, b As Double) As Double
        Return a + b
    End Function

    Private Sub MostrarSuma(a As Double, b As Double)
        lblResultado.Text = "Suma: " & (a + b)
    End Sub

    Private Sub btnEjecutar_Click(sender As Object, e As EventArgs) Handles
btnEjecutar.Click
        lblResultado.Text = "Resultado Function: " & Sumar(5, 3)
        MostrarSuma(5, 3)
    End Sub
End Class
```

PROGRAMA 4: Paso de parámetros por valor

Enunciado: Cambiar un valor dentro de un `Function` usando `ByVal`.

```
Public Class Form1
    Private Function Incrementar(ByVal x As Integer) As Integer
        x += 10
        Return x
    End Function

    Private Sub btnCalcular_Click(sender As Object, e As EventArgs) Handles
btnCalcular.Click
        Dim numero As Integer = 5
        lblResultado.Text = "Antes: 5, Después: " & Incrementar(numero)
    End Sub
End Class
```

Explicación: `ByVal` no modifica la variable original fuera de la función.

PROGRAMA 5: Paso de parámetros por referencia

Enunciado: Cambiar un valor dentro de un `Sub` usando `ByRef`.

```
Public Class Form1
    Private Sub Incrementar(ByRef x As Integer)
        x += 10
    End Sub

    Private Sub btnCalcular_Click(sender As Object, e As EventArgs) Handles
btnCalcular.Click
        Dim numero As Integer = 5
        Incrementar(numero)
        lblResultado.Text = "Valor modificado: " & numero
    End Sub
End Class
```

Explicación: `ByRef` modifica la variable original.

PROGRAMA 6: Función integrada Math.Sqrt

Enunciado: Calcular la raíz cuadrada de un número usando `Math.Sqrt`.

```
Public Class Form1
    Private Sub btnCalcular_Click(sender As Object, e As EventArgs) Handles
btnCalcular.Click
        Dim numero As Double = CDbl(txtNumero.Text)
        lblResultado.Text = "Raíz cuadrada: " & Math.Sqrt(numero)
    End Sub
End Class
```

PROGRAMA 7: Función integrada Math.Pow

Enunciado: Elevar un número a una potencia usando `Math.Pow`.

```
Public Class Form1
    Private Sub btnCalcular_Click(sender As Object, e As EventArgs) Handles
btnCalcular.Click
        Dim base As Double = CDbl(txtBase.Text)
        Dim exponente As Double = CDbl(txtExponente.Text)
        lblResultado.Text = "Potencia: " & Math.Pow(base, exponente)
    End Sub
End Class
```

PROGRAMA 8: Función integrada String.Length

Enunciado: Contar caracteres de un `TextBox`.

```
Public Class Form1
    Private Sub btnContar_Click(sender As Object, e As EventArgs) Handles
btnContar.Click
        lblResultado.Text = "Número de caracteres: " & txtTexto.Text.Length
    End Sub
End Class
```

PROGRAMA 9: Función integrada UCase y LCase

Enunciado: Convertir texto a mayúsculas y minúsculas.

```
Public Class Form1
    Private Sub btnConvertir_Click(sender As Object, e As EventArgs) Handles
btnConvertir.Click
        lblMayus.Text = txtTexto.Text.ToUpper()
        lblMinus.Text = txtTexto.Text.ToLower()
    End Sub
End Class
```

PROGRAMA 10: Función recursiva factorial

Enunciado: Calcular factorial de un número usando recursión.

```
Public Class Form1
    Private Function Factorial(n As Integer) As Integer
        If n <= 1 Then
            Return 1
        Else
            Return n * Factorial(n - 1)
        End If
    End Function

    Private Sub btnCalcular_Click(sender As Object, e As EventArgs) Handles
btnCalcular.Click
        Dim numero As Integer = CInt(txtNumero.Text)
        lblResultado.Text = "Factorial: " & Factorial(numero)
    End Sub
End Class
```

PROGRAMA 11: Función recursiva Fibonacci

Enunciado: Calcular un término de Fibonacci recursivamente.

```
Public Class Form1
    Private Function Fibonacci(n As Integer) As Integer
        If n <= 1 Then
            Return n
        Else
            Return Fibonacci(n - 1) + Fibonacci(n - 2)
        End If
    End Function

    Private Sub btnCalcular_Click(sender As Object, e As EventArgs) Handles
btnCalcular.Click
        Dim n As Integer = CInt(txtNumero.Text)
        lblResultado.Text = "Fibonacci: " & Fibonacci(n)
    End Sub
End Class
```

PROGRAMA 12: Sumar números usando Function

Enunciado: Crear función que sume dos números y devuelva resultado.

```
Public Class Form1
    Private Function Sumar(a As Double, b As Double) As Double
        Return a + b
    End Function

    Private Sub btnCalcular_Click(sender As Object, e As EventArgs) Handles
btnCalcular.Click
        lblResultado.Text = "Suma: " & Sumar(CDbl(txtA.Text), CDbl(txtB.Text))
    End Sub
End Class
```

PROGRAMA 13: Mostrar mensaje con Sub

Enunciado: Crear Sub que muestre saludo personalizado.

```
Public Class Form1
    Private Sub Saludar(nombre As String)
        lblResultado.Text = "Hola, " & nombre
    End Sub

    Private Sub btnSaludar_Click(sender As Object, e As EventArgs) Handles
btnSaludar.Click
        Saludar(txtNombre.Text)
    End Sub
End Class
```

PROGRAMA 14: Intercambiar valores con ByRef

Enunciado: Intercambiar dos números usando Sub y `ByRef`.

```
Public Class Form1
    Private Sub Intercambiar(ByRef a As Integer, ByRef b As Integer)
        Dim temp As Integer = a
        a = b
        b = temp
    End Sub

    Private Sub btnIntercambiar_Click(sender As Object, e As EventArgs) Handles
btnIntercambiar.Click
        Dim x As Integer = CInt(txtX.Text)
        Dim y As Integer = CInt(txtY.Text)
        Intercambiar(x, y)
        lblResultado.Text = "X: " & x & " Y: " & y
    End Sub
End Class
```

PROGRAMA 15: Contar vocales con Function

Enunciado: Contar cuántas vocales tiene un texto.

```
Public Class Form1
    Private Function ContarVocales(texto As String) As Integer
        Dim contador As Integer = 0
        For Each c As Char In texto.ToLower()
            If "aeiou".Contains(c) Then contador += 1
        Next
        Return contador
    End Function

    Private Sub btnContar_Click(sender As Object, e As EventArgs) Handles
btnContar.Click
        lblResultado.Text = "Vocales: " & ContarVocales(txtTexto.Text)
    End Sub
End Class
```

PROGRAMA 16: Validar edad con Function

Enunciado: Validar si la persona es mayor de edad.

```
Public Class Form1
    Private Function EsMayor(edad As Integer) As Boolean
        Return edad >= 18
    End Function

    Private Sub btnValidar_Click(sender As Object, e As EventArgs) Handles
btnValidar.Click
        Dim edad As Integer = CInt(txtEdad.Text)
        lblResultado.Text = If(EsMayor(edad), "Mayor de edad", "Menor de edad")
    End Sub
End Class
```

PROGRAMA 17: Generar tabla de multiplicar (Sub)

Enunciado: Mostrar tabla de multiplicar de un número en un `ListBox`.

```
Public Class Form1
    Private Sub TablaMultiplicar(num As Integer)
        lstTabla.Items.Clear()
        For i As Integer = 1 To 10
            lstTabla.Items.Add(num & " x " & i & " = " & (num * i))
        Next
    End Sub

    Private Sub btnGenerar_Click(sender As Object, e As EventArgs) Handles
btnGenerar.Click
        TablaMultiplicar(CInt(txtNumero.Text))
    End Sub
End Class
```

PROGRAMA 18: Número par o impar (Function)

Enunciado: Determinar si un número es par o impar.

```
Public Class Form1
    Private Function EsPar(n As Integer) As Boolean
        Return n Mod 2 = 0
    End Function

    Private Sub btnVerificar_Click(sender As Object, e As EventArgs) Handles
btnVerificar.Click
        Dim n As Integer = CInt(txtNumero.Text)
        lblResultado.Text = If(EsPar(n), "Par", "Impar")
    End Sub
End Class
```

PROGRAMA 19: Suma recursiva de números naturales

Enunciado: Calcular suma de 1 a N recursivamente.

```
Public Class Form1
    Private Function SumaRecursiva(n As Integer) As Integer
        If n = 1 Then
            Return 1
        Else
            Return n + SumaRecursiva(n - 1)
        End If
    End Function

    Private Sub btnCalcular_Click(sender As Object, e As EventArgs) Handles
btnCalcular.Click
        lblResultado.Text = "Suma: " & SumaRecursiva(CInt(txtNumero.Text))
    End Sub
End Class
```

PROGRAMA 20: Invertir texto recursivamente

Enunciado: Invertir un texto usando función recursiva.

```
Public Class Form1
    Private Function Invertir(texto As String) As String
        If texto.Length <= 1 Then
            Return texto
        Else
            Return Invertir(texto.Substring(1)) & texto(0)
        End If
    End Function

    Private Sub btnInvertir_Click(sender As Object, e As EventArgs) Handles
btnInvertir.Click
        lblResultado.Text = Invertir(txtTexto.Text)
    End Sub
End Class
```

Ejercicios

16.1 Uso de MsgBox

Programa 1 – Mensaje simple de información

Enunciado: Muestra un mensaje simple con `MsgBox`.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        MsgBox("Hola, este es un mensaje simple de información.",
MsgBoxStyle.Information, "Mensaje")
    End Sub
End Class
```

Programa 2 – Mensaje de advertencia

Enunciado: Muestra un mensaje de advertencia usando `MsgBoxStyle.Exclamation`.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        MsgBox("Cuidado, esto es una advertencia.", MsgBoxStyle.Exclamation,
"Advertencia")
    End Sub
End Class
```

Programa 3 – Mensaje de pregunta con opciones Sí/No

Enunciado: Pregunta al usuario si desea continuar y captura la respuesta.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim respuesta As MsgBoxResult
        respuesta = MsgBox("¿Deseas continuar?", MsgBoxStyle.YesNo +
MsgBoxStyle.Question, "Pregunta")
        If respuesta = MsgBoxResult.Yes Then
            MsgBox("Elegiste continuar.")
        Else
            MsgBox("Elegiste no continuar.")
        End If
    End Sub
End Class
```

16.2 MessageBox con botones y estilos

Programa 4 – MessageBox con OK y Cancelar

Enunciado: Muestra un `MessageBox` con botones OK y Cancelar y detecta la elección.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim resultado As DialogResult
        resultado = MessageBox.Show("¿Deseas salir?", "Confirmar salida",
MessageBoxButtons.OKCancel, MessageBoxIcon.Question)
        If resultado = DialogResult.OK Then
            MsgBox("Has seleccionado OK")
        Else
            MsgBox("Has seleccionado Cancelar")
        End If
    End Sub
End Class
```

Programa 5 – MessageBox con Sí/No/Cancelar

Enunciado: Permite elegir entre Sí, No o Cancelar.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim resultado As DialogResult
        resultado = MessageBox.Show("¿Deseas guardar los cambios?", "Guardar
cambios", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Warning)
        Select Case resultado
            Case DialogResult.Yes
                MsgBox("Has elegido Sí")
            Case DialogResult.No
                MsgBox("Has elegido No")
            Case DialogResult.Cancel
                MsgBox("Has elegido Cancelar")
        End Select
    End Sub
End Class
```

Programa 6 – MessageBox con íconos de información

Enunciado: Muestra un `MessageBox` con icono de información y solo botón OK.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        MessageBox.Show("Operación completada correctamente.", "Información",
MessageBoxButtons.OK, MessageBoxIcon.Information)
    End Sub
End Class
```

Programa 7 – MessageBox con ícono de error

Enunciado: Muestra un mensaje de error.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        MessageBox.Show("Ha ocurrido un error inesperado.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
    End Sub
End Class
```

Programa 8 – MessageBox con ícono de advertencia

Enunciado: Muestra un mensaje de advertencia con botón OK.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        MessageBox.Show("Se detectó un problema potencial.", "Advertencia",
MessageBoxButtons.OK, MessageBoxIcon.Warning)
    End Sub
End Class
```

Programa 9 – MessageBox con opción personalizada

Enunciado: Permite al usuario seleccionar entre botones personalizados y actúa según la selección.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim resultado As DialogResult
        resultado = MessageBox.Show("Selecciona una opción", "Opciones",
MessageBoxButtons.YesNo, MessageBoxIcon.Question)
        If resultado = DialogResult.Yes Then
            MsgBox("Elegiste la primera opción")
        Else
            MsgBox("Elegiste la segunda opción")
        End If
    End Sub
End Class
```

16.3 Cuadros de diálogo comunes

Programa 10 – OpenFileDialog

Enunciado: Permite al usuario seleccionar un archivo y muestra la ruta seleccionada.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim openFile As New OpenFileDialog
        If openFile.ShowDialog() = DialogResult.OK Then
            MsgBox("Archivo seleccionado: " & openFile.FileName)
        End If
    End Sub
End Class
```

Programa 11 – SaveFileDialog

Enunciado: Permite al usuario elegir dónde guardar un archivo y muestra la ruta.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim saveFile As New SaveFileDialog
        If saveFile.ShowDialog() = DialogResult.OK Then
            MsgBox("Guardar archivo en: " & saveFile.FileName)
        End If
    End Sub
End Class
```

Programa 12 – ColorDialog

Enunciado: Permite al usuario seleccionar un color y cambia el color de fondo del formulario.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim colorDialog As New ColorDialog
        If colorDialog.ShowDialog() = DialogResult.OK Then
            Me.BackColor = colorDialog.Color
        End If
    End Sub
End Class
```

Programa 13 – FontDialog

Enunciado: Permite al usuario seleccionar una fuente y cambiarla en un Label.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim fontDialog As New FontDialog
        If fontDialog.ShowDialog() = DialogResult.OK Then
            Label1.Font = fontDialog.Font
        End If
    End Sub
End Class
```

Programa 14 – OpenFileDialog con filtro

Enunciado: Abre solo archivos de texto usando filtro.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim openFile As New OpenFileDialog
        openFile.Filter = "Archivos de texto (*.txt)|*.txt"
        If openFile.ShowDialog() = DialogResult.OK Then
            MsgBox("Archivo seleccionado: " & openFile.FileName)
        End If
    End Sub
End Class
```

Programa 15 – SaveFileDialog con extensión predeterminada

Enunciado: Guarda archivo predeterminando extensión `.txt`.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim saveFile As New SaveFileDialog
        saveFile.DefaultExt = "txt"
        saveFile.Filter = "Archivos de texto (*.txt)|*.txt"
        If saveFile.ShowDialog() = DialogResult.OK Then
            MsgBox("Archivo guardado en: " & saveFile.FileName)
        End If
    End Sub
End Class
```

Programa 16 – ColorDialog para texto de Label

Enunciado: Permite seleccionar un color y aplicarlo a un Label.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim colorDialog As New ColorDialog
        If colorDialog.ShowDialog() = DialogResult.OK Then
            Label1.ForeColor = colorDialog.Color
        End If
    End Sub
End Class
```

Programa 17 – FontDialog con vista previa

Enunciado: Permite seleccionar fuente con vista previa en Label.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim fontDialog As New FontDialog
        fontDialog.ShowColor = True
        fontDialog.Font = Label1.Font
        fontDialog.Color = Label1.ForeColor
        If fontDialog.ShowDialog() = DialogResult.OK Then
            Label1.Font = fontDialog.Font
            Label1.ForeColor = fontDialog.Color
        End If
    End Sub
End Class
```

Programa 18 – OpenFileDialog múltiples archivos

Enunciado: Permite seleccionar varios archivos y mostrar sus nombres.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim openFile As New OpenFileDialog
        openFile.Multiselect = True
        If openFile.ShowDialog() = DialogResult.OK Then
            Dim archivos As String = String.Join(", ", openFile.FileNames)
            MsgBox("Archivos seleccionados: " & archivos)
        End If
    End Sub
End Class
```

Programa 19 – SaveFileDialog con verificación

Enunciado: Verifica si el archivo existe antes de guardar.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim saveFile As New SaveFileDialog
        If saveFile.ShowDialog() = DialogResult.OK Then
            If My.Computer.FileSystem.FileExists(saveFile.FileName) Then
                MsgBox("El archivo ya existe, será reemplazado.")
            End If
            MsgBox("Archivo guardado en: " & saveFile.FileName)
        End If
    End Sub
End Class
```

Programa 20 – ColorDialog y mensaje final

Enunciado: Cambia el color de fondo y muestra un mensaje de confirmación.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim colorDialog As New ColorDialog
        If colorDialog.ShowDialog() = DialogResult.OK Then
            Me.BackColor = colorDialog.Color
            MsgBox("Color de fondo cambiado correctamente.")
        End If
    End Sub
End Class
```

Ejercicios

17.1. Manipulación de cadenas (String, StringBuilder)

Programa 1: Concatenación de cadenas

Enunciado: Une dos cadenas ingresadas por el usuario y muestra el resultado.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim cadena1 As String = TextBox1.Text
        Dim cadena2 As String = TextBox2.Text
        Dim resultado As String = cadena1 & " " & cadena2
        MessageBox.Show("Resultado: " & resultado)
    End Sub
End Class
```

Programa 2: Longitud de cadena

Enunciado: Muestra la longitud de una cadena ingresada.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim texto As String = TextBox1.Text
        MessageBox.Show("Longitud: " & texto.Length)
    End Sub
End Class
```

Programa 3: Convertir a mayúsculas y minúsculas

Enunciado: Convierte el texto ingresado a mayúsculas y minúsculas.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim texto As String = TextBox1.Text
        MessageBox.Show("Mayúsculas: " & texto.ToUpper() & vbCrLf & "Minúsculas:
" & texto.ToLower())
    End Sub
End Class
```

Programa 4: Uso de StringBuilder

Enunciado: Modifica dinámicamente una cadena con StringBuilder.

```
Imports System.Text
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim sb As New StringBuilder("Hola")
        sb.Append(" Mundo")
        sb.Replace("Mundo", "VB.NET")
        MessageBox.Show(sb.ToString())
    End Sub
End Class
```

Programa 5: Buscar y reemplazar texto

Enunciado: Reemplaza todas las apariciones de una palabra por otra.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim texto As String = TextBox1.Text
        Dim resultado As String = texto.Replace("perro", "gato")
        MessageBox.Show("Resultado: " & resultado)
    End Sub
End Class
```

17.2. Formato de cadenas y números

Programa 6: Formateo de números con decimales

Enunciado: Muestra un número con dos decimales.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim num As Double = CDbl(TextBox1.Text)
        MessageBox.Show("Número formateado: " & num.ToString("F2"))
    End Sub
End Class
```

Programa 7: Formato de moneda

Enunciado: Muestra un número como moneda local.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim num As Double = CDbl(TextBox1.Text)
        MessageBox.Show("Moneda: " & num.ToString("C2"))
    End Sub
End Class
```

Programa 8: Formato de porcentaje

Enunciado: Convierte un número decimal en porcentaje.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim num As Double = CDbl(TextBox1.Text)
        MessageBox.Show("Porcentaje: " & num.ToString("P"))
    End Sub
End Class
```

Programa 9: Alinear texto con formato

Enunciado: Muestra un texto alineado a la derecha y a la izquierda.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim texto As String = TextBox1.Text
        MessageBox.Show("Izquierda: " & texto.PadRight(20) & vbCrLf & "Derecha:
" & texto.PadLeft(20))
    End Sub
End Class
```

Programa 10: Formato de fechas

Enunciado: Muestra la fecha actual en formato largo y corto.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim hoy As Date = Date.Now
        MessageBox.Show("Fecha corta: " & hoy.ToShortDateString() & vbCrLf &
"Fecha larga: " & hoy.ToLongDateString())
    End Sub
End Class
```

17.3. Funciones de conversión de datos (CInt, CDbl, CStr, etc.)

Programa 11: Convertir texto a entero

Enunciado: Convierte el texto ingresado a número entero y suma 10.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim valor As Integer = CInt(TextBox1.Text)
        MessageBox.Show("Resultado: " & (valor + 10))
    End Sub
End Class
```

Programa 12: Convertir texto a doble

Enunciado: Convierte texto a número decimal y calcula la raíz cuadrada.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim num As Double = CDbl(TextBox1.Text)
        MessageBox.Show("Raíz cuadrada: " & Math.Sqrt(num))
    End Sub
End Class
```

Programa 13: Convertir número a cadena

Enunciado: Convierte un número ingresado a cadena y muestra mensaje personalizado.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim num As Double = CDbl(TextBox1.Text)
        MessageBox.Show("El número ingresado es: " & CStr(num))
    End Sub
End Class
```

Programa 14: Conversión de entero a booleano

Enunciado: Convierte un número a booleano y muestra True si es distinto de cero.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim valor As Integer = CInt(TextBox1.Text)
        Dim estado As Boolean = CBool(valor)
        MessageBox.Show("Estado: " & estado)
    End Sub
End Class
```

Programa 15: Conversión de cadena a fecha

Enunciado: Convierte un texto en formato de fecha a objeto Date.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim fecha As Date = CDate(TextBox1.Text)
        MessageBox.Show("Día de la semana: " & fecha.DayOfWeek)
    End Sub
End Class
```

17.4. Funciones de validación

Programa 16: Validar si el texto es numérico

Enunciado: Verifica si el contenido de TextBox es un número.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        If IsNumeric(TextBox1.Text) Then
            MessageBox.Show("Es un número")
        Else
            MessageBox.Show("No es un número")
        End If
    End Sub
End Class
```

Programa 17: Validar si el texto es alfabético

Enunciado: Verifica que solo contenga letras.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        If TextBox1.Text.All(AddressOf Char.IsLetter) Then
            MessageBox.Show("Solo contiene letras")
        Else
            MessageBox.Show("Contiene otros caracteres")
        End If
    End Sub
End Class
```

Programa 18: Validar longitud mínima de texto

Enunciado: Verifica que el texto tenga al menos 5 caracteres.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        If TextBox1.Text.Length >= 5 Then
            MessageBox.Show("Texto válido")
        Else
            MessageBox.Show("Texto demasiado corto")
        End If
    End Sub
End Class
```

Programa 19: Validar si la entrada es fecha

Enunciado: Comprueba si el contenido es una fecha válida.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim fecha As Date
        If Date.TryParse(TextBox1.Text, fecha) Then
            MessageBox.Show("Fecha válida")
        Else
            MessageBox.Show("Fecha inválida")
        End If
    End Sub
End Class
```

Programa 20: Validar si el texto no está vacío

Enunciado: Verifica que el usuario haya ingresado algún valor.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        If String.IsNullOrWhiteSpace(TextBox1.Text) Then
            MessageBox.Show("El texto está vacío")
        Else
            MessageBox.Show("Texto ingresado: " & TextBox1.Text)
        End If
    End Sub
End Class
```

Ejercicios

Programa 1: Declaración de variable Date y mostrarla

Enunciado: Este programa declara una variable de tipo `Date`, le asigna la fecha actual y la muestra en un Label.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim fechaActual As Date = Date.Now
        Label1.Text = "Fecha actual: " & fechaActual.ToString()
    End Sub
End Class
```

Programa 2: Fecha y hora actual en TextBox

Enunciado: Declara una variable `Date` y muestra la fecha y hora actual en un TextBox.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim ahora As Date = DateTime.Now
        TextBox1.Text = ahora.ToString()
    End Sub
End Class
```

Programa 3: Propiedades de DateTime

Enunciado: Muestra día, mes, año, hora, minuto y segundo de la fecha actual usando propiedades de `DateTime`.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim ahora As Date = DateTime.Now
        Label1.Text = "Día: " & ahora.Day & vbCrLf &
                      "Mes: " & ahora.Month & vbCrLf &
                      "Año: " & ahora.Year & vbCrLf &
                      "Hora: " & ahora.Hour & vbCrLf &
                      "Minuto: " & ahora.Minute & vbCrLf &
                      "Segundo: " & ahora.Second
    End Sub
End Class
```

Programa 4: Sumar días a una fecha

Enunciado: Suma 10 días a la fecha actual y muestra el resultado.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim fechaActual As Date = DateTime.Now
        Dim fechaFutura As Date = fechaActual.AddDays(10)
        Label1.Text = "Fecha futura: " & fechaFutura.ToShortDateString()
    End Sub
End Class
```

Programa 5: Restar fechas

Enunciado: Calcula la diferencia en días entre dos fechas.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim fechaInicio As Date = #9/1/2025#
        Dim fechaFin As Date = #9/27/2025#
        Dim diferencia As TimeSpan = fechaFin - fechaInicio
        Label1.Text = "Diferencia en días: " & diferencia.Days
    End Sub
End Class
```

Programa 6: Comparar fechas

Enunciado: Compara dos fechas y muestra cuál es mayor.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim fecha1 As Date = #9/1/2025#
        Dim fecha2 As Date = #9/27/2025#
        If fecha1 > fecha2 Then
            Label1.Text = "Fecha1 es posterior a Fecha2"
        ElseIf fecha1 < fecha2 Then
            Label1.Text = "Fecha1 es anterior a Fecha2"
        Else
            Label1.Text = "Ambas fechas son iguales"
        End If
    End Sub
End Class
```

Programa 7: Formato de fecha corta

Enunciado: Muestra la fecha actual en formato corto `dd/MM/yyyy`.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim fechaActual As Date = DateTime.Now
        Label1.Text = fechaActual.ToShortDateString()
    End Sub
End Class
```

Programa 8: Formato de fecha larga

Enunciado: Muestra la fecha actual en formato largo `miércoles, 27 de septiembre de 2025`.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim fechaActual As Date = DateTime.Now
        Label1.Text = fechaActual.ToLongDateString()
    End Sub
End Class
```

Programa 9: Formato de hora

Enunciado: Muestra la hora actual en formato `HH:mm:ss`.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim horaActual As Date = DateTime.Now
        Label1.Text = horaActual.ToLongTimeString()
    End Sub
End Class
```

Programa 10: Convertir String a Date

Enunciado: Convierte un texto con fecha a tipo Date y muestra el resultado.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim textoFecha As String = "27/09/2025"
        Dim fechaConvertida As Date = CDate(textoFecha)
        Label1.Text = "Fecha convertida: " & fechaConvertida.ToShortDateString()
    End Sub
End Class
```

Programa 11: Sumar meses a una fecha

Enunciado: Suma 3 meses a la fecha actual.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim fechaActual As Date = DateTime.Now
        Dim nuevaFecha As Date = fechaActual.AddMonths(3)
        Label1.Text = "Fecha dentro de 3 meses: " &
nuevaFecha.ToShortDateString()
    End Sub
End Class
```

Programa 12: Sumar años a una fecha

Enunciado: Calcula la fecha dentro de 5 años.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim fechaActual As Date = DateTime.Now
        Dim fechaFutura As Date = fechaActual.AddYears(5)
        Label1.Text = "Fecha dentro de 5 años: " &
fechaFutura.ToShortDateString()
    End Sub
End Class
```

Programa 13: Sumar horas a la hora actual

Enunciado: Agrega 2 horas a la hora actual.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim horaActual As Date = DateTime.Now
        Dim horaFutura As Date = horaActual.AddHours(2)
        Label1.Text = "Hora dentro de 2 horas: " & horaFutura.ToLongTimeString()
    End Sub
End Class
```

Programa 14: Día de la semana

Enunciado: Muestra el nombre del día de la semana de la fecha actual.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim fechaActual As Date = DateTime.Now
        Label1.Text = "Hoy es: " & fechaActual.DayOfWeek.ToString()
    End Sub
End Class
```

Programa 15: Diferencia en horas

Enunciado: Calcula la diferencia en horas entre dos fechas.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim fecha1 As Date = #9/27/2025 08:00#
        Dim fecha2 As Date = #9/27/2025 15:30#
        Dim diferencia As TimeSpan = fecha2 - fecha1
        Label1.Text = "Diferencia en horas: " & diferencia.TotalHours
    End Sub
End Class
```

Programa 16: Diferencia en minutos

Enunciado: Calcula la diferencia en minutos entre dos fechas.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim fecha1 As Date = #9/27/2025 08:00#
        Dim fecha2 As Date = #9/27/2025 08:45#
        Dim diferencia As TimeSpan = fecha2 - fecha1
        Label1.Text = "Diferencia en minutos: " & diferencia.TotalMinutes
    End Sub
End Class
```

Programa 17: Mostrar fecha en formato personalizado

Enunciado: Muestra la fecha en formato `dd-MM-yyyy`.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim fechaActual As Date = DateTime.Now
        Label1.Text = fechaActual.ToString("dd-MM-yyyy")
    End Sub
End Class
```

Programa 18: Validar si texto es fecha

Enunciado: Valida si un texto ingresado en TextBox es una fecha correcta.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim texto As String = TextBox1.Text
        Dim fecha As Date
        If Date.TryParse(texto, fecha) Then
            Label1.Text = "Texto válido: " & fecha.ToShortDateString()
        Else
            Label1.Text = "Texto no es una fecha válida"
        End If
    End Sub
End Class
```

Programa 19: Día del año

Enunciado: Muestra en qué día del año estamos.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim fechaActual As Date = DateTime.Now
        Label1.Text = "Hoy es el día número: " & fechaActual.DayOfYear
    End Sub
End Class
```

Programa 20: Día hábil siguiente

Enunciado: Calcula el siguiente día hábil sumando días y omitiendo fines de semana.

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim fecha As Date = DateTime.Now
        Do
            fecha = fecha.AddDays(1)
        Loop While fecha.DayOfWeek = DayOfWeek.Saturday Or fecha.DayOfWeek =
DayOfWeek.Sunday
        Label1.Text = "Siguiente día hábil: " & fecha.ToShortDateString()
    End Sub
End Class
```

---

19. Creación y Uso de Múltiples Formularios

En una aplicación de Windows Forms, es común trabajar con más de un formulario. Por ejemplo, un formulario principal (menú) que abre formularios secundarios (configuración, reportes, etc.). En este capítulo veremos cómo abrir y cerrar formularios, la diferencia entre formularios modales y no modales, cómo pasar datos entre ellos y cómo acceder a propiedades y métodos de un formulario desde otro.

19.1. Abrir y cerrar formularios secundarios

Para agregar un nuevo formulario en **Visual Studio**:

1. Haz clic derecho en el proyecto → **Agregar** → **Windows Form**.
2. Asigna un nombre (por ejemplo, `FormSecundario.vb`).

Abrir un formulario secundario:

```
' Desde el Formulario Principal
Dim frm As New FormSecundario()
frm.Show()
```

Cerrar un formulario:

```
Me.Close()   ' Cierra el formulario actual
```

Ejemplo práctico:

En el **FormPrincipal**, colocar un botón `btnAbrir` con este código:

```
Private Sub btnAbrir_Click(sender As Object, e As EventArgs) Handles
btnAbrir.Click
    Dim frm As New FormSecundario()
    frm.Show()
End Sub
```

Y en el **FormSecundario**, un botón `btnCerrar`:

```
Private Sub btnCerrar_Click(sender As Object, e As EventArgs) Handles
btnCerrar.Click
    Me.Close()
End Sub
```

19.2. Formularios modales y no modales

- **Formulario no modal (`Show`)**: permite seguir interactuando con otros formularios.
- **Formulario modal (`ShowDialog`)**: bloquea la interacción hasta que se cierre.

Ejemplo de formulario modal:

```
Dim frm As New FormSecundario()
frm.ShowDialog()   ' El usuario debe cerrarlo antes de volver al principal
```

Esto es útil para cuadros de configuración, inicio de sesión o confirmaciones.

19.3. Pasar datos entre formularios

Existen varias formas de enviar información entre formularios.

1. Usando el constructor del formulario secundario:

```
' FormSecundario.vb
Public Class FormSecundario
    Private mensaje As String

    Public Sub New(texto As String)
        InitializeComponent()
        mensaje = texto
    End Sub

    Private Sub FormSecundario_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        MessageBox.Show("Dato recibido: " & mensaje)
    End Sub
End Class

' FormPrincipal.vb
Dim frm As New FormSecundario("Hola desde el Formulario Principal")
frm.Show()
```

---

2. Usando propiedades públicas:

```
' FormSecundario.vb
Public Class FormSecundario
    Public Property Mensaje As String

    Private Sub FormSecundario_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        Label1.Text = Mensaje
    End Sub
End Class

' FormPrincipal.vb
Dim frm As New FormSecundario()
frm.Mensaje = "Texto enviado desde el Principal"
frm.Show()
```

19.4. Uso de propiedades y métodos entre formularios

Un formulario puede exponer **propiedades y métodos** que otros formularios pueden usar.

Ejemplo con propiedad:

```
' FormSecundario.vb
Public Class FormSecundario
    Public Property NombreUsuario As String
End Class

' FormPrincipal.vb
Dim frm As New FormSecundario()
frm.NombreUsuario = "Ana Pérez"
frm.Show()
```

Ejemplo con método:

```
' FormSecundario.vb
Public Class FormSecundario
    Public Sub MostrarMensaje(texto As String)
        MessageBox.Show("Mensaje recibido: " & texto)
    End Sub
End Class

' FormPrincipal.vb
Dim frm As New FormSecundario()
frm.Show()
frm.MostrarMensaje("Hola desde el formulario principal")
```

Comunicación entre Controles de Distintos Formularios

20.1. Enviar valores de TextBox de un formulario a otro

En aplicaciones Windows Forms con Visual Basic .NET, es muy común que necesitemos enviar información introducida en un `TextBox` de un formulario a otro formulario. Esto puede hacerse de varias maneras, como:

- Pasar datos mediante constructores.
- Usar propiedades públicas en los formularios.
- Definir variables compartidas o módulos globales.

Ejemplo 1: Pasar datos mediante propiedades públicas

Supongamos que tenemos dos formularios: **Form1** y **Form2**. En `Form1` el usuario escribe su nombre en un `TextBox`, y al abrir `Form2`, ese nombre se mostrará en un `Label`.

Form1.vb

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        ' Crear instancia del segundo formulario
        Dim f2 As New Form2()
        ' Asignar el valor del TextBox a la propiedad pública de Form2
        f2.Nombre = TextBox1.Text
        f2.Show()
    End Sub
End Class
```

Form2.vb

```
Public Class Form2
    ' Propiedad pública para recibir datos desde Form1
    Public Property Nombre As String

    Private Sub Form2_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        ' Mostrar el valor recibido en un Label
        Label1.Text = "Hola, " & Nombre
    End Sub
End Class
```

Este método es limpio porque evita dependencias innecesarias y sigue el principio de encapsulación.

20.2. Actualizar controles en formularios abiertos

En muchas aplicaciones, necesitamos actualizar en tiempo real los controles de un formulario que ya está abierto desde otro formulario. Para ello, debemos mantener una referencia al formulario objetivo en lugar de crear uno nuevo.

Ejemplo: Actualizar un Label en un formulario ya abierto

Supongamos que tenemos un formulario principal (**FormPrincipal**) y otro secundario (**FormSecundario**). El secundario contiene un `TextBox` y un `Button` que, al pulsarse, actualizarán un `Label` en el formulario principal.

FormPrincipal.vb

```
Public Class FormPrincipal

    Public Sub ActualizarLabel(texto As String)

        Label1.Text = texto

    End Sub


    Private Sub ButtonAbrir_Click(sender As Object, e As EventArgs) Handles
ButtonAbrir.Click

        Dim f2 As New FormSecundario(Me) ' Pasamos la referencia del principal

        f2.Show()

    End Sub

End Class
```

FormSecundario.vb

```
Public Class FormSecundario

    Private principal As FormPrincipal


    ' Constructor que recibe el formulario principal

    Public Sub New(frm As FormPrincipal)

        InitializeComponent()

        principal = frm

    End Sub


    Private Sub ButtonEnviar_Click(sender As Object, e As EventArgs) Handles
ButtonEnviar.Click

        ' Llamamos al método público del formulario principal

        principal.ActualizarLabel(TextBox1.Text)

    End Sub

End Class
```

En este caso, la comunicación se hace directamente invocando un método público en el formulario principal. Esto permite **actualizar controles de un formulario abierto en tiempo real**.

20.3. Formularios MDI (Multiple Document Interface)

Un **formulario MDI (Multiple Document Interface)** es un formulario que actúa como contenedor de otros formularios hijos. Este patrón se utiliza mucho en programas como editores de texto o software de diseño, donde varias ventanas de documentos pueden estar abiertas dentro de una principal.

Creación de un formulario MDI

1. Crear un formulario llamado **FormMDI**.
2. Cambiar su propiedad **IsMdiContainer** a `True`.
3. Crear otros formularios hijos (**FormHijo1**, **FormHijo2**, etc.).
4. Abrir los formularios hijos dentro del contenedor.

FormMDI.vb

```
Public Class FormMDI

    Private Sub AbrirHijo1ToolStripMenuItem_Click(sender As Object, e As
EventArgs) Handles AbrirHijo1ToolStripMenuItem.Click

        Dim hijo1 As New FormHijo1()

        hijo1.MdiParent = Me

        hijo1.Show()

    End Sub


    Private Sub AbrirHijo2ToolStripMenuItem_Click(sender As Object, e As
EventArgs) Handles AbrirHijo2ToolStripMenuItem.Click

        Dim hijo2 As New FormHijo2()

        hijo2.MdiParent = Me

        hijo2.Show()

    End Sub

End Class
```

Comunicación entre formularios MDI

En un entorno MDI, los formularios hijos pueden comunicarse entre sí a través del contenedor.

Ejemplo: Supongamos que **FormHijo1** tiene un `TextBox` y queremos enviar su valor a un `Label` en **FormHijo2**.

FormHijo1.vb

```
Public Class FormHijo1

    Private Sub ButtonEnviar_Click(sender As Object, e As EventArgs) Handles
ButtonEnviar.Click

        ' Buscar si FormHijo2 está abierto

        For Each frm As Form In Me.MdiParent.MdiChildren

            If TypeOf frm Is FormHijo2 Then

                DirectCast(frm, FormHijo2).ActualizarLabel(TextBox1.Text)

            End If

        Next

    End Sub

End Class
```

FormHijo2.vb

```
Public Class FormHijo2

    Public Sub ActualizarLabel(texto As String)

        Label1.Text = texto

    End Sub

End Class
```

De esta forma, podemos **enviar datos entre formularios hijos MDI** usando el contenedor como puente.

Ejercicios

Programa 19.1 – Abrir un formulario secundario simple

Enunciado: Este programa muestra cómo abrir un segundo formulario desde el formulario principal.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim frm As New Form2
        frm.Show() ' Abrir formulario secundario
    End Sub
End Class

' Form2.vb
Public Class Form2
    Private Sub Form2_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Me.Text = "Formulario Secundario"
    End Sub
End Class
```

Programa 19.2 – Cerrar un formulario secundario

Enunciado: Este programa abre un formulario secundario y lo cierra mediante un botón.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim frm As New Form2
        frm.Show()
    End Sub
End Class

' Form2.vb
Public Class Form2
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Me.Close() ' Cierra este formulario
    End Sub
End Class
```

Programa 19.3 – Abrir formulario modal

Enunciado: Se abre un formulario secundario de manera modal, bloqueando el formulario principal hasta que se cierre.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim frm As New Form2
        frm.ShowDialog() ' Formulario modal
    End Sub
End Class

' Form2.vb
Public Class Form2
    Private Sub Form2_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Me.Text = "Formulario Modal"
    End Sub
End Class
```

Programa 19.4 – Abrir formulario no modal

Enunciado: Se abre un formulario secundario no modal, permitiendo interactuar con el principal.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim frm As New Form2
        frm.Show() ' Formulario no modal
    End Sub
End Class
```

Programa 19.5 – Pasar datos del formulario principal al secundario

Enunciado: Se envía un valor desde el Form1 a Form2 usando el constructor.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim mensaje As String = "Hola desde Form1"
        Dim frm As New Form2(mensaje)
        frm.Show()
    End Sub
End Class

' Form2.vb
Public Class Form2
    Public Sub New(mensaje As String)
        InitializeComponent()
        Label1.Text = mensaje
    End Sub
End Class
```

Programa 19.6 – Pasar datos del formulario secundario al principal

Enunciado: Se devuelve información del Form2 al Form1 usando propiedades.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim frm As New Form2
        If frm.ShowDialog() = DialogResult.OK Then
            TextBox1.Text = frm.Respuesta
        End If
    End Sub
End Class

' Form2.vb
Public Class Form2
    Public Property Respuesta As String
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Respuesta = TextBox1.Text
        Me.DialogResult = DialogResult.OK
        Me.Close()
    End Sub
End Class
```

Programa 19.7 – Uso de métodos de formulario secundario

Enunciado: Se llama un método público del Form2 desde Form1.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim frm As New Form2
        frm.Show()
        frm.CambiarColor()
    End Sub
End Class

' Form2.vb
Public Class Form2
    Public Sub CambiarColor()
        Me.BackColor = Color.Aqua
    End Sub
End Class
```

Programa 19.8 – Uso de propiedades de formulario secundario

Enunciado: Se modifica el título del formulario secundario desde el principal mediante una propiedad pública.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim frm As New Form2
        frm.Titulo = "Nuevo Título desde Form1"
        frm.Show()
    End Sub
End Class

' Form2.vb
Public Class Form2
    Public Property Titulo As String
        Get
            Return Me.Text
        End Get
        Set(value As String)
            Me.Text = value
        End Set
    End Property
End Class
```

Programa 19.9 – Abrir varios formularios secundarios

Enunciado: Se abre más de un formulario secundario a la vez.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim frm1 As New Form2
        Dim frm2 As New Form2
        frm1.Show()
        frm2.Show()
    End Sub
End Class
```

Programa 19.10 – Contar cuántos formularios secundarios están abiertos

Enunciado: Se lleva un registro de los formularios abiertos.

```
' Form1.vb
Public Class Form1
    Private formulariosAbiertos As Integer = 0

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim frm As New Form2
        AddHandler frm.FormClosed, AddressOf FormClosedHandler
        formulariosAbiertos += 1
        frm.Show()
        Label1.Text = "Formularios abiertos: " & formulariosAbiertos
    End Sub

    Private Sub FormClosedHandler(sender As Object, e As FormClosedEventArgs)
        formulariosAbiertos -= 1
        Label1.Text = "Formularios abiertos: " & formulariosAbiertos
    End Sub
End Class
```

Programa 19.11 – Abrir formulario secundario con valores iniciales

Enunciado: Se envían varios valores al Form2 mediante propiedades públicas.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim frm As New Form2
        frm.Nombre = "Ana"
        frm.Edad = 25
        frm.Show()
    End Sub
End Class

' Form2.vb
Public Class Form2
    Public Property Nombre As String
    Public Property Edad As Integer

    Private Sub Form2_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Label1.Text = "Nombre: " & Nombre & ", Edad: " & Edad
    End Sub
End Class
```

Programa 19.12 – Validar datos en formulario secundario

Enunciado: Form2 valida datos antes de cerrarse.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim frm As New Form2
        frm.ShowDialog()
    End Sub
End Class

' Form2.vb
Public Class Form2
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        If TextBox1.Text = "" Then
            MessageBox.Show("Debe ingresar un valor")
        Else
            Me.Close()
        End If
    End Sub
End Class
```

Programa 19.13 – Usar formulario secundario como diálogo de confirmación

Enunciado: Form2 actúa como confirmación (Sí/No).

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim frm As New Form2
        If frm.ShowDialog() = DialogResult.Yes Then
            MessageBox.Show("Usuario confirmó")
        End If
    End Sub
End Class

' Form2.vb
Public Class Form2
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Me.DialogResult = DialogResult.Yes
        Me.Close()
    End Sub

    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles
Button2.Click
        Me.DialogResult = DialogResult.No
        Me.Close()
    End Sub
End Class
```

Programa 19.14 – Actualizar datos del principal desde secundario

Enunciado: Form2 cambia datos de Form1 usando referencia.

```
' Form1.vb
Public Class Form1
    Public Sub ActualizarLabel(texto As String)
        Label1.Text = texto
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim frm As New Form2(Me)
        frm.Show()
    End Sub
End Class

' Form2.vb
Public Class Form2
    Private frmPrincipal As Form1
    Public Sub New(frm As Form1)
        InitializeComponent()
        frmPrincipal = frm
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        frmPrincipal.ActualizarLabel("Texto cambiado desde Form2")
        Me.Close()
    End Sub
End Class
```

Programa 19.15 – Abrir formulario secundario con control dinámico

Enunciado: Form2 crea un control en tiempo de ejecución.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim frm As New Form2
        frm.Show()
    End Sub
End Class

' Form2.vb
Public Class Form2
    Private Sub Form2_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim btn As New Button
        btn.Text = "Click dinámico"
        btn.Location = New Point(50, 50)
        AddHandler btn.Click, AddressOf Btn_Click
        Me.Controls.Add(btn)
    End Sub

    Private Sub Btn_Click(sender As Object, e As EventArgs)
        MessageBox.Show("Botón creado dinámicamente")
    End Sub
End Class
```

Programa 19.16 – Formularios con menú que abre secundarios

Enunciado: Menú principal abre diferentes formularios.

```
' Form1.vb
Public Class Form1
    Private Sub Formulario1ToolStripMenuItem_Click(sender As Object, e As
EventArgs) Handles Formulario1ToolStripMenuItem.Click
        Dim frm As New Form2
        frm.Show()
    End Sub
    Private Sub Formulario2ToolStripMenuItem_Click(sender As Object, e As
EventArgs) Handles Formulario2ToolStripMenuItem.Click
        Dim frm As New Form3
        frm.Show()
    End Sub
End Class
```

Programa 19.17 – Abrir formulario secundario maximizado

Enunciado: Form2 se abre en pantalla completa.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim frm As New Form2
        frm.WindowState = FormWindowState.Maximized
        frm.Show()
    End Sub
End Class
```

Programa 19.18 – Abrir formulario secundario minimizado

Enunciado: Form2 se abre minimizado en la barra de tareas.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim frm As New Form2
        frm.WindowState = FormWindowState.Minimized
        frm.Show()
    End Sub
End Class
```

Programa 19.19 – Evitar abrir múltiples veces el mismo formulario

Enunciado: Se comprueba si el formulario ya está abierto antes de abrirlo de nuevo.

```
' Form1.vb
Public Class Form1
    Private frm As Form2 = Nothing

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        If frm Is Nothing OrElse frm.IsDisposed Then
            frm = New Form2
            frm.Show()
        Else
            frm.BringToFront()
        End If
    End Sub
End Class
```

Programa 19.20 – Formularios con comunicación bidireccional

Enunciado: Form1 y Form2 intercambian datos en tiempo real mediante eventos.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim frm As New Form2
        AddHandler frm.DatosCambiados, AddressOf ActualizarLabel
        frm.Show()
    End Sub

    Private Sub ActualizarLabel(texto As String)
        Label1.Text = texto
    End Sub
End Class

' Form2.vb
Public Class Form2
    Public Event DatosCambiados(ByVal texto As String)

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        RaiseEvent DatosCambiados("Cambio desde Form2")
    End Sub
End Class
```

Ejercicios

Programa 20.1 – Enviar valor de un TextBox a otro formulario

Enunciado: Este programa muestra cómo enviar el texto de un `TextBox` en el formulario principal a otro `TextBox` en un formulario secundario al presionar un botón.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim f2 As New Form2()
        f2.TextBox1.Text = Me.TextBox1.Text
        f2.Show()
    End Sub
End Class

' Form2.vb
Public Class Form2
    ' Contiene TextBox1 para recibir el texto de Form1
End Class
```

Programa 20.2 – Actualizar TextBox en formulario abierto

Enunciado: Este programa permite que un formulario secundario actualice un `TextBox` del formulario principal.

```
' Form1.vb
Public Class Form1
    Public Sub ActualizarTexto(texto As String)
        TextBox1.Text = texto
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim f2 As New Form2(Me)
        f2.Show()
    End Sub
End Class

' Form2.vb
Public Class Form2
    Private fPrincipal As Form1

    Public Sub New(frm As Form1)
        InitializeComponent()
        fPrincipal = frm
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        fPrincipal.ActualizarTexto(TextBox1.Text)
    End Sub
End Class
```

Programa 20.3 – Formularios MDI (Multiple Document Interface)

Enunciado: Este programa crea un formulario principal MDI y permite abrir múltiples formularios hijos dentro de él.

```
' FormMDI.vb
Public Class FormMDI
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim hijo As New FormHijo()
        hijo.MdiParent = Me
        hijo.Show()
    End Sub
End Class

' FormHijo.vb
Public Class FormHijo
    ' Formulario hijo vacío
End Class
```

Programa 20.4 – Pasar valores entre TextBox de formularios con propiedades

Enunciado: Este programa utiliza propiedades públicas para pasar valores de un formulario a otro.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim f2 As New Form2()
        f2.ValorRecibido = TextBox1.Text
        f2.Show()
    End Sub
End Class

' Form2.vb
Public Class Form2
    Public Property ValorRecibido As String
        Get
            Return TextBox1.Text
        End Get
        Set(value As String)
            TextBox1.Text = value
        End Set
    End Property
End Class
```

Programa 20.5 – Actualizar Label en formulario abierto desde otro

Enunciado: Este programa permite actualizar el texto de un `Label` en un formulario secundario desde el principal.

```
' Form1.vb
Public Class Form1
    Private f2 As New Form2()

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        f2.Label1.Text = TextBox1.Text
        f2.Show()
    End Sub
End Class

' Form2.vb
Public Class Form2
    ' Contiene Label1
End Class
```

Programa 20.6 – Enviar datos numéricos entre formularios

Enunciado: Este programa permite enviar un número de un `TextBox` a otro formulario y sumarle 10.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim f2 As New Form2()
        f2.RecibirNumero(CInt(TextBox1.Text))
        f2.Show()
    End Sub
End Class

' Form2.vb
Public Class Form2
    Public Sub RecibirNumero(num As Integer)
        TextBox1.Text = (num + 10).ToString()
    End Sub
End Class
```

Programa 20.7 – Formulario secundario cierra formulario principal

Enunciado: Este programa permite que un formulario secundario cierre el formulario principal al presionar un botón.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim f2 As New Form2(Me)
        f2.Show()
    End Sub
End Class

' Form2.vb
Public Class Form2
    Private fPrincipal As Form1

    Public Sub New(frm As Form1)
        InitializeComponent()
        fPrincipal = frm
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        fPrincipal.Close()
    End Sub
End Class
```

Programa 20.8 – Enviar lista de datos entre formularios

Enunciado: Este programa envía los valores de un `ListBox` en Form1 a otro `ListBox` en Form2.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim f2 As New Form2()
        For Each item As String In ListBox1.Items
            f2.ListBox1.Items.Add(item)
        Next
        f2.Show()
    End Sub
End Class

' Form2.vb
Public Class Form2
    ' Contiene ListBox1
End Class
```

Programa 20.9 – Comunicación bidireccional

Enunciado: Este programa permite que dos formularios intercambien datos mediante propiedades públicas.

```
' Form1.vb
Public Class Form1
    Private f2 As New Form2()

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        f2.TextBox1.Text = TextBox1.Text
        f2.Show()
    End Sub

    Public Property ValorRecibido As String
        Get
            Return TextBox1.Text
        End Get
        Set(value As String)
            TextBox1.Text = value
        End Set
    End Property
End Class

' Form2.vb
Public Class Form2
    Public Property ValorParaForm1 As String
End Class
```

Programa 20.10 – Actualizar DataGridView en otro formulario

Enunciado: Este programa permite enviar filas desde un `DataGridView` en Form1 a otro `DataGridView` en Form2.

```
' Form1.vb

Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click

        Dim f2 As New Form2()

        For Each row As DataGridViewRow In DataGridView1.Rows

            If Not row.IsNewRow Then

                f2.DataGridView1.Rows.Add(row.Cells(0).Value,
row.Cells(1).Value)

            End If

        Next

        f2.Show()

    End Sub

End Class


' Form2.vb

Public Class Form2

    ' Contiene DataGridView1

End Class
```

Programa 20.11 – Enviar valor al abrir formulario modal

Enunciado: Este programa abre un formulario modal (`ShowDialog`) y envía un valor para mostrarlo en un `TextBox`.

```
' Form1.vb

Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click

        Dim f2 As New Form2()

        f2.Valor = TextBox1.Text

        f2.ShowDialog()

    End Sub

End Class


' Form2.vb

Public Class Form2

    Public Property Valor As String

        Get

            Return TextBox1.Text

        End Get

        Set(value As String)

            TextBox1.Text = value

        End Set

    End Property

End Class
```

Programa 20.12 – Formulario secundario actualiza ListBox principal

Enunciado: Permite que un formulario secundario agregue elementos a un `ListBox` en el formulario principal.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim f2 As New Form2(Me)
        f2.Show()
    End Sub

    Public Sub AgregarElemento(texto As String)
        ListBox1.Items.Add(texto)
    End Sub
End Class

' Form2.vb
Public Class Form2
    Private fPrincipal As Form1

    Public Sub New(frm As Form1)
        InitializeComponent()
        fPrincipal = frm
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        fPrincipal.AgregarElemento(TextBox1.Text)
    End Sub
End Class
```

Programa 20.13 – Formulario MDI con menú para abrir hijos

Enunciado: Formulario MDI con menú que permite abrir varios formularios hijos de manera independiente.

```
' FormMDI.vb

Public Class FormMDI

    Private Sub NuevoToolStripMenuItem_Click(sender As Object, e As EventArgs)
Handles NuevoToolStripMenuItem.Click

        Dim hijo As New FormHijo()

        hijo.MdiParent = Me

        hijo.Show()

    End Sub

End Class


' FormHijo.vb

Public Class FormHijo

    ' Formulario hijo vacío

End Class
```

Programa 20.14 – Pasar ComboBox seleccionado

Enunciado: Este programa envía el valor seleccionado de un `ComboBox` en Form1 a Form2.

```
' Form1.vb

Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click

        Dim f2 As New Form2()

        f2.ComboBox1.SelectedItem = ComboBox1.SelectedItem

        f2.Show()

    End Sub

End Class


' Form2.vb

Public Class Form2

    ' Contiene ComboBox1

End Class
```

Programa 20.15 – Comunicación usando eventos personalizados

Enunciado: Se utiliza un evento personalizado para actualizar un control en otro formulario.

```
' Form1.vb

Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click

        Dim f2 As New Form2()

        AddHandler f2.TextoEnviado, AddressOf ActualizarLabel

        f2.Show()

    End Sub


    Private Sub ActualizarLabel(texto As String)

        Label1.Text = texto

    End Sub

End Class


' Form2.vb

Public Class Form2

    Public Event TextoEnviado(texto As String)


    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click

        RaiseEvent TextoEnviado(TextBox1.Text)

    End Sub

End Class
```

Programa 20.16 – Formulario secundario cambia color de Form1

Enunciado: Este programa permite que un formulario secundario cambie el color de fondo del formulario principal.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim f2 As New Form2(Me)
        f2.Show()
    End Sub

    Public Sub CambiarColor(c As Color)
        Me.BackColor = c
    End Sub
End Class

' Form2.vb
Public Class Form2
    Private fPrincipal As Form1

    Public Sub New(frm As Form1)
        InitializeComponent()
        fPrincipal = frm
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        fPrincipal.CambiarColor(Color.LightBlue)
    End Sub
End Class
```

Programa 20.17 – Formulario secundario modifica ProgressBar de Form1

Enunciado: Se actualiza una `ProgressBar` en Form1 desde Form2.

```
' Form1.vb

Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click

        Dim f2 As New Form2(Me)

        f2.Show()

    End Sub


    Public Sub ActualizarProgress(valor As Integer)

        ProgressBar1.Value = valor

    End Sub

End Class


' Form2.vb

Public Class Form2

    Private fPrincipal As Form1


    Public Sub New(frm As Form1)

        InitializeComponent()

        fPrincipal = frm

    End Sub


    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click

        fPrincipal.ActualizarProgress(CInt(TextBox1.Text))

    End Sub

End Class
```

Programa 20.18 – Enviar CheckBox marcado entre formularios

Enunciado: Este programa permite enviar el estado de un `CheckBox` de Form1 a Form2.

```
' Form1.vb

Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click

        Dim f2 As New Form2()

        f2.CheckBox1.Checked = CheckBox1.Checked

        f2.Show()

    End Sub

End Class


' Form2.vb

Public Class Form2

    ' Contiene CheckBox1

End Class
```

Programa 20.19 – Actualizar ListView de Form1 desde Form2

Enunciado: Este programa permite agregar ítems a un `ListView` en Form1 desde un formulario secundario.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim f2 As New Form2(Me)
        f2.Show()
    End Sub

    Public Sub AgregarItem(item As String)
        ListView1.Items.Add(item)
    End Sub
End Class

' Form2.vb
Public Class Form2
    Private fPrincipal As Form1

    Public Sub New(frm As Form1)
        InitializeComponent()
        fPrincipal = frm
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        fPrincipal.AgregarItem(TextBox1.Text)
    End Sub
End Class
```

Programa 20.20 – Formulario secundario modifica propiedad de control en Form1

Enunciado: Este programa permite cambiar propiedades de un control (como `Enabled`) en el formulario principal desde un formulario secundario.

```
' Form1.vb
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        Dim f2 As New Form2(Me)
        f2.Show()
    End Sub

    Public Sub DeshabilitarBoton()
        Button1.Enabled = False
    End Sub
End Class

' Form2.vb
Public Class Form2
    Private fPrincipal As Form1

    Public Sub New(frm As Form1)
        InitializeComponent()
        fPrincipal = frm
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
        fPrincipal.DeshabilitarBoton()
    End Sub
End Class
```

Parte VII – Temas Avanzados y Proyectos

---

21. Manejo de Archivos y Directorios

El manejo de archivos es uno de los temas más importantes en aplicaciones de escritorio. Muchas veces necesitamos guardar datos para que persistan aunque se cierre el programa, o bien leer información desde archivos existentes.

En Visual Basic .NET disponemos de varias clases en el **espacio de nombres `System.IO`**, como:

- **File** y **FileInfo**: permiten crear, copiar, mover y borrar archivos.
- **Directory** y **DirectoryInfo**: permiten manipular directorios (carpetas).
- **StreamReader** y **StreamWriter**: permiten leer y escribir archivos de texto.
- **BinaryReader** y **BinaryWriter**: permiten trabajar con archivos binarios.

A continuación, veremos ejemplos prácticos para Windows Forms.

21.1. Lectura y escritura de archivos de texto

Los archivos de texto son los más comunes para almacenar datos simples. Visual Basic .NET permite leer y escribir en ellos de forma sencilla.

Ejemplo: Guardar texto de un TextBox en un archivo

Form1.vb

```
Imports System.IO

Public Class Form1
    Private Sub ButtonGuardar_Click(sender As Object, e As EventArgs) Handles
ButtonGuardar.Click
        ' Guardar el contenido del TextBox en un archivo
        File.WriteAllText("C:\temp\archivo.txt", TextBox1.Text)
        MessageBox.Show("Archivo guardado correctamente")
    End Sub
End Class
```

Ejemplo: Leer el contenido de un archivo en un TextBox

```
Private Sub ButtonLeer_Click(sender As Object, e As EventArgs) Handles
ButtonLeer.Click
    If File.Exists("C:\temp\archivo.txt") Then
        TextBox1.Text = File.ReadAllText("C:\temp\archivo.txt")
    Else
        MessageBox.Show("El archivo no existe")
    End If
End Sub
```

Con `File.ReadAllText` y `File.WriteAllText` simplificamos mucho las operaciones.

21.2. Uso de StreamReader y StreamWriter

Cuando queremos más control sobre la lectura y escritura, usamos **StreamReader** y **StreamWriter**. Estas clases permiten leer línea por línea o escribir progresivamente.

Ejemplo: Escribir en un archivo línea a línea con StreamWriter

```
Private Sub ButtonEscribir_Click(sender As Object, e As EventArgs) Handles
ButtonEscribir.Click
    Using escritor As New StreamWriter("C:\temp\lista.txt", True) ' True =
añadir al final
        escritor.WriteLine(TextBox1.Text)
    End Using
    MessageBox.Show("Texto añadido al archivo")
End Sub
```

Ejemplo: Leer un archivo línea a línea con StreamReader

```
Private Sub ButtonLeerLineas_Click(sender As Object, e As EventArgs) Handles
ButtonLeerLineas.Click
    If File.Exists("C:\temp\lista.txt") Then
        Using lector As New StreamReader("C:\temp\lista.txt")
            Dim linea As String
            TextBox1.Clear()
            While Not lector.EndOfStream
                linea = lector.ReadLine()
                TextBox1.AppendText(linea & Environment.NewLine)
            End While
        End Using
    Else
        MessageBox.Show("El archivo no existe")
    End If
End Sub
```

Con este método podemos procesar grandes archivos sin necesidad de cargarlos completos en memoria.

21.3. Manejo de archivos binarios

Los archivos binarios permiten guardar datos de manera estructurada y más eficiente que el texto. En lugar de caracteres, se almacenan los valores en formato binario.

En Visual Basic .NET utilizamos:

- **BinaryWriter**: para escribir datos binarios.
- **BinaryReader**: para leer datos binarios.

Ejemplo: Guardar datos binarios

```
Private Sub ButtonGuardarBinario_Click(sender As Object, e As EventArgs) Handles
ButtonGuardarBinario.Click
    Using bw As New BinaryWriter(File.Open("C:\temp\datos.bin",
FileMode.Create))
        bw.Write(TextBox1.Text)       ' Guardar una cadena
        bw.Write(CInt(TextBox2.Text)) ' Guardar un número entero
        bw.Write(DateTime.Now.ToString()) ' Guardar fecha actual
    End Using
    MessageBox.Show("Datos guardados en binario")
```

```
End Sub
```

Ejemplo: Leer datos binarios

```
Private Sub ButtonLeerBinario_Click(sender As Object, e As EventArgs) Handles
ButtonLeerBinario.Click
    If File.Exists("C:\temp\datos.bin") Then
        Using br As New BinaryReader(File.Open("C:\temp\datos.bin",
FileMode.Open))
            Dim texto As String = br.ReadString()
            Dim numero As Integer = br.ReadInt32()
            Dim fecha As String = br.ReadString()

            MessageBox.Show("Texto: " & texto & vbCrLf &
                            "Número: " & numero & vbCrLf &
                            "Fecha: " & fecha)
        End Using
    Else
        MessageBox.Show("El archivo binario no existe")
    End If
End Sub
```

Este ejemplo muestra cómo guardar y recuperar distintos tipos de datos de manera binaria.

Capítulo 22 – Manejo de Errores y Depuración

Introducción

El manejo de errores es una parte fundamental en cualquier aplicación. Ningún programa está libre de fallos, ya sean provocados por datos incorrectos, archivos inexistentes, conexiones de red fallidas o errores de lógica.

En Visual Basic .NET, el tratamiento de errores se realiza principalmente mediante el uso de **excepciones** y estructuras de control como `Try...Catch...Finally`.

Además, Visual Studio ofrece potentes **herramientas de depuración** que permiten encontrar errores de manera más rápida y efectiva.

En este capítulo aprenderás:

1. Cómo manejar errores con `Try, Catch, Finally`.
2. Cómo crear excepciones personalizadas.
3. Cómo usar las herramientas de depuración de Visual Studio.

22.1. Try, Catch, Finally

La estructura básica para manejar errores en Visual Basic .NET es:

```
Try
    ' Código que puede producir un error
Catch ex As Exception
    ' Código que se ejecuta si ocurre una excepción
Finally
    ' Código que siempre se ejecuta (haya o no error)
End Try
```

Ejemplo 1: División entre cero

```
Private Sub ButtonDividir_Click(sender As Object, e As EventArgs) Handles
ButtonDividir.Click
    Try
        Dim num1 As Integer = CInt(TextBox1.Text)
        Dim num2 As Integer = CInt(TextBox2.Text)
        Dim resultado As Integer = num1 \ num2
        LabelResultado.Text = "Resultado: " & resultado
    Catch ex As DivideByZeroException
        MessageBox.Show("No se puede dividir entre cero", "Error")
    Catch ex As Exception
        MessageBox.Show("Error inesperado: " & ex.Message)
    Finally
        TextBox1.Clear()
        TextBox2.Clear()
    End Try
End Sub
```

Explicación:

- Si el usuario introduce un `0`, el programa captura la excepción `DivideByZeroException`.
- Si ocurre otro error (texto en lugar de número), lo captura el `Catch ex As Exception`.
- El bloque `Finally` limpia los `TextBox` siempre, ocurra o no un error.

22.2. Excepciones personalizadas

Además de las excepciones estándar, podemos crear nuestras propias excepciones para controlar situaciones específicas de nuestra aplicación.

Ejemplo: Crear una excepción personalizada

```
' Clase para excepción personalizada
Public Class EdadInvalidaException
    Inherits ApplicationException

    Public Sub New(mensaje As String)
        MyBase.New(mensaje)
    End Sub
End Class
```

Ejemplo: Usar la excepción personalizada

```
Private Sub ButtonValidarEdad_Click(sender As Object, e As EventArgs) Handles
ButtonValidarEdad.Click
    Try
        Dim edad As Integer = CInt(TextBoxEdad.Text)

        If edad < 0 Or edad > 120 Then
            Throw New EdadInvalidaException("La edad ingresada no es válida")
        End If

        MessageBox.Show("Edad aceptada: " & edad)

    Catch ex As EdadInvalidaException
        MessageBox.Show(ex.Message, "Error de Validación")
    Catch ex As Exception
        MessageBox.Show("Error inesperado: " & ex.Message)
    End Try
End Sub
```

Ventajas de las excepciones personalizadas:

- Mejoran la claridad del código.
- Permiten diferenciar entre errores comunes y errores específicos de la aplicación.
- Facilitan la depuración y el mantenimiento.

22.3. Herramientas de depuración en Visual Studio

Visual Studio incluye un depurador muy potente que permite analizar paso a paso la ejecución del programa y detectar errores.

Principales herramientas de depuración

1. **Breakpoints (Puntos de interrupción):**
   Permiten detener la ejecución en una línea específica de código. Se colocan con clic en el margen izquierdo del editor o con la tecla **F9**.
2. **Step Into (F11):**
   Ejecuta el código línea por línea, entrando en los métodos llamados.
3. **Step Over (F10):**
   Ejecuta la línea actual sin entrar dentro de los métodos.
4. **Step Out (Shift + F11):**
   Sale del método actual y vuelve al llamador.
5. **Ventanas de depuración:**
   - **Autos y Locals:** muestran las variables en el contexto actual.
   - **Watch:** permite observar valores específicos de variables durante la ejecución.
   - **Call Stack:** muestra la pila de llamadas, es decir, qué métodos fueron llamados hasta el punto actual.
6. **Inmediato:**
   Permite ejecutar comandos o evaluar expresiones en tiempo de depuración.

Ejemplo práctico con breakpoints

1. Coloca un breakpoint en la línea donde haces la división en el ejemplo de 22.1.
2. Ejecuta el programa en modo **Debug (F5)**.
3. Introduce números en los `TextBox` y pulsa el botón.
4. El programa se detendrá en el breakpoint.
5. Con **F10 o F11** avanza paso a paso y observa cómo cambian los valores de las variables.

Esto permite encontrar errores lógicos antes de que se conviertan en fallos graves.

Capítulo 23 – Proyectos Prácticos con Windows Forms

Introducción

En este capítulo vamos a desarrollar proyectos prácticos usando Windows Forms, aplicando los conocimientos adquiridos en capítulos anteriores. Cada proyecto se enfoca en diferentes controles y funcionalidades, desde operaciones matemáticas hasta aplicaciones con menús, bases de datos y multimedia.

23.1. Calculadora con TextBox, Label y Button

Descripción

Una calculadora básica que permite sumar, restar, multiplicar y dividir dos números introducidos en `TextBox`.

Controles

- **TextBox1, TextBox2**: entrada de números.
- **ButtonSuma, ButtonResta, ButtonMultiplicar, ButtonDividir**: operaciones.
- **LabelResultado**: muestra el resultado.

Código

```
Public Class FormCalculadora
    Private Sub ButtonSuma_Click(sender As Object, e As EventArgs) Handles
ButtonSuma.Click
        Dim a As Double = CDbl(TextBox1.Text)
        Dim b As Double = CDbl(TextBox2.Text)
        LabelResultado.Text = "Resultado: " & (a + b).ToString()
    End Sub

    Private Sub ButtonResta_Click(sender As Object, e As EventArgs) Handles
ButtonResta.Click
        Dim a As Double = CDbl(TextBox1.Text)
        Dim b As Double = CDbl(TextBox2.Text)
        LabelResultado.Text = "Resultado: " & (a - b).ToString()
    End Sub

    Private Sub ButtonMultiplicar_Click(sender As Object, e As EventArgs)
Handles ButtonMultiplicar.Click
        Dim a As Double = CDbl(TextBox1.Text)
        Dim b As Double = CDbl(TextBox2.Text)
        LabelResultado.Text = "Resultado: " & (a * b).ToString()
    End Sub

    Private Sub ButtonDividir_Click(sender As Object, e As EventArgs) Handles
ButtonDividir.Click
        Try
            Dim a As Double = CDbl(TextBox1.Text)
            Dim b As Double = CDbl(TextBox2.Text)
            LabelResultado.Text = "Resultado: " & (a / b).ToString()
```

```
        Catch ex As DivideByZeroException
            MessageBox.Show("No se puede dividir entre cero")
        End Try
    End Sub
End Class
```

23.2. Agenda de contactos con interacción entre formularios

Descripción

Un programa con dos formularios:

- **FormAgenda** muestra la lista de contactos.
- **FormNuevoContacto** permite añadir un nuevo contacto.

Controles

- **ListBoxContactos (FormAgenda)**.
- **TextBoxNombre, TextBoxTelefono, ButtonGuardar (FormNuevoContacto)**.

Código

FormAgenda.vb

```
Public Class FormAgenda
    Public Sub AgregarContacto(nombre As String, telefono As String)
        ListBoxContactos.Items.Add(nombre & " - " & telefono)
    End Sub

    Private Sub ButtonNuevo_Click(sender As Object, e As EventArgs) Handles
ButtonNuevo.Click
        Dim nuevo As New FormNuevoContacto(Me)
        nuevo.Show()
    End Sub
End Class
```

FormNuevoContacto.vb

```
Public Class FormNuevoContacto
    Private agenda As FormAgenda

    Public Sub New(frm As FormAgenda)
        InitializeComponent()
        agenda = frm
    End Sub

    Private Sub ButtonGuardar_Click(sender As Object, e As EventArgs) Handles
ButtonGuardar.Click
        agenda.AgregarContacto(TextBoxNombre.Text, TextBoxTelefono.Text)
        Me.Close()
    End Sub
End Class
```

23.3. Aplicación de inventario con DataGridView

Descripción

Una aplicación que gestiona productos en inventario usando un `DataGridView`.

Controles

- **DataGridView1** con columnas: Producto, Cantidad, Precio.
- **TextBoxProducto, TextBoxCantidad, TextBoxPrecio, ButtonAgregar**.

Código

```
Public Class FormInventario
    Private Sub FormInventario_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        DataGridView1.Columns.Add("Producto", "Producto")
        DataGridView1.Columns.Add("Cantidad", "Cantidad")
        DataGridView1.Columns.Add("Precio", "Precio")
    End Sub

    Private Sub ButtonAgregar_Click(sender As Object, e As EventArgs) Handles
ButtonAgregar.Click
        DataGridView1.Rows.Add(TextBoxProducto.Text, TextBoxCantidad.Text,
TextBoxPrecio.Text)
    End Sub
End Class
```

23.4. Bloc de notas con menús y ToolStrip

Descripción

Un bloc de notas simple con opciones para abrir, guardar y editar texto.

Controles

- **MenuStrip** con opciones: Archivo (Nuevo, Abrir, Guardar, Salir).
- **ToolStrip** con botones de acceso rápido.
- **TextBoxMultiline** para el texto.

Código

```
Imports System.IO

Public Class FormBlocNotas
    Private archivoActual As String = ""

    Private Sub NuevoToolStripMenuItem_Click(sender As Object, e As EventArgs)
Handles NuevoToolStripMenuItem.Click
        TextBox1.Clear()
        archivoActual = ""
    End Sub
```

```
    Private Sub AbrirToolStripMenuItem_Click(sender As Object, e As EventArgs)
Handles AbrirToolStripMenuItem.Click
        Dim dlg As New OpenFileDialog()
        dlg.Filter = "Archivos de texto|*.txt"
        If dlg.ShowDialog() = DialogResult.OK Then
            TextBox1.Text = File.ReadAllText(dlg.FileName)
            archivoActual = dlg.FileName
        End If
    End Sub

    Private Sub GuardarToolStripMenuItem_Click(sender As Object, e As EventArgs)
Handles GuardarToolStripMenuItem.Click
        If archivoActual = "" Then
            Dim dlg As New SaveFileDialog()
            dlg.Filter = "Archivos de texto|*.txt"
            If dlg.ShowDialog() = DialogResult.OK Then
                archivoActual = dlg.FileName
            End If
        End If
        If archivoActual <> "" Then
            File.WriteAllText(archivoActual, TextBox1.Text)
        End If
    End Sub

    Private Sub SalirToolStripMenuItem_Click(sender As Object, e As EventArgs)
Handles SalirToolStripMenuItem.Click
        Me.Close()
    End Sub
End Class
```

23.5. Reproductor multimedia con botones y diálogos

Descripción

Un reproductor básico que permite abrir y reproducir archivos de audio y video.

Controles

- **AxWindowsMediaPlayer1 (ActiveX)**.
- **ButtonAbrir, ButtonPlay, ButtonPausa, ButtonStop**.

Código

```
Imports AxWMPLib

Public Class FormReproductor
    Private archivo As String

    Private Sub ButtonAbrir_Click(sender As Object, e As EventArgs) Handles
ButtonAbrir.Click
        Dim dlg As New OpenFileDialog()
        dlg.Filter = "Archivos multimedia|*.mp3;*.mp4;*.wav;*.avi"
        If dlg.ShowDialog() = DialogResult.OK Then
            archivo = dlg.FileName
            AxWindowsMediaPlayer1.URL = archivo
        End If
    End Sub
```

```
    Private Sub ButtonPlay_Click(sender As Object, e As EventArgs) Handles
ButtonPlay.Click
        AxWindowsMediaPlayer1.Ctlcontrols.play()
    End Sub

    Private Sub ButtonPausa_Click(sender As Object, e As EventArgs) Handles
ButtonPausa.Click
        AxWindowsMediaPlayer1.Ctlcontrols.pause()
    End Sub

    Private Sub ButtonStop_Click(sender As Object, e As EventArgs) Handles
ButtonStop.Click
        AxWindowsMediaPlayer1.Ctlcontrols.stop()
    End Sub
End Class
```

Ejercicios

Programa 1 – Crear un archivo de texto y escribir contenido

Enunciado: Este programa permite al usuario crear un archivo de texto y escribir contenido en él desde un TextBox.
Explicación: Usamos `StreamWriter` para crear y escribir en un archivo de texto.

```
Imports System.IO

Public Class Form1
    Private Sub btnGuardar_Click(sender As Object, e As EventArgs) Handles
btnGuardar.Click
        Dim ruta As String = "C:\archivo1.txt"
        Using sw As New StreamWriter(ruta)
            sw.WriteLine(txtContenido.Text)
        End Using
        MessageBox.Show("Archivo creado y guardado correctamente.")
    End Sub
End Class
```

Programa 2 – Leer un archivo de texto

Enunciado: Este programa permite al usuario seleccionar un archivo de texto y mostrar su contenido en un TextBox.
Explicación: Usamos `StreamReader` para leer línea por línea el archivo.

```
Imports System.IO

Public Class Form1
    Private Sub btnLeer_Click(sender As Object, e As EventArgs) Handles
btnLeer.Click
        Dim ruta As String = "C:\archivo1.txt"
        Using sr As New StreamReader(ruta)
            txtContenido.Text = sr.ReadToEnd()
        End Using
    End Sub
End Class
```

Programa 3 – Agregar texto a un archivo existente

Enunciado: Permite agregar texto a un archivo ya existente sin sobrescribir su contenido.
Explicación: `StreamWriter` con el parámetro `True` permite la escritura en modo append.

```
Imports System.IO

Public Class Form1
    Private Sub btnAgregar_Click(sender As Object, e As EventArgs) Handles
btnAgregar.Click
        Dim ruta As String = "C:\archivo1.txt"
        Using sw As New StreamWriter(ruta, True)
            sw.WriteLine(txtContenido.Text)
        End Using
        MessageBox.Show("Texto agregado correctamente.")
    End Sub
End Class
```

Programa 4 – Copiar un archivo

Enunciado: Este programa copia un archivo de una ruta a otra.
Explicación: Usamos `File.Copy` de la clase `System.IO`.

```
Imports System.IO

Public Class Form1
    Private Sub btnCopiar_Click(sender As Object, e As EventArgs) Handles
btnCopiar.Click
        File.Copy("C:\archivo1.txt", "C:\archivo2.txt", True)
        MessageBox.Show("Archivo copiado correctamente.")
    End Sub
End Class
```

Programa 5 – Mover un archivo

Enunciado: Permite mover un archivo de una ubicación a otra.
Explicación: `File.Move` traslada el archivo a la ruta destino.

```
Imports System.IO

Public Class Form1
    Private Sub btnMover_Click(sender As Object, e As EventArgs) Handles
btnMover.Click
        File.Move("C:\archivo2.txt", "D:\archivo2.txt")
        MessageBox.Show("Archivo movido correctamente.")
    End Sub
End Class
```

Programa 6 – Eliminar un archivo

Enunciado: Permite eliminar un archivo especificado por el usuario.
Explicación: `File.Delete` elimina el archivo.

```
Imports System.IO

Public Class Form1
    Private Sub btnEliminar_Click(sender As Object, e As EventArgs) Handles
btnEliminar.Click
        File.Delete("D:\archivo2.txt")
        MessageBox.Show("Archivo eliminado.")
    End Sub
End Class
```

Programa 7 – Crear un directorio

Enunciado: Permite crear un directorio en una ruta específica.
Explicación: `Directory.CreateDirectory` crea la carpeta si no existe.

```
Imports System.IO

Public Class Form1
    Private Sub btnCrearDir_Click(sender As Object, e As EventArgs) Handles
btnCrearDir.Click
        Directory.CreateDirectory("C:\MiDirectorio")
        MessageBox.Show("Directorio creado.")
    End Sub
End Class
```

Programa 8 – Eliminar un directorio

Enunciado: Elimina un directorio y todos sus contenidos.
Explicación: `Directory.Delete` con el parámetro `True` elimina subcarpetas y archivos.

```
Imports System.IO

Public Class Form1
    Private Sub btnEliminarDir_Click(sender As Object, e As EventArgs) Handles
btnEliminarDir.Click
        Directory.Delete("C:\MiDirectorio", True)
        MessageBox.Show("Directorio eliminado.")
    End Sub
End Class
```

Programa 9 – Listar archivos de un directorio

Enunciado: Muestra todos los archivos de un directorio en un ListBox.
Explicación: `Directory.GetFiles` obtiene un arreglo de archivos.

```
Imports System.IO

Public Class Form1
    Private Sub btnListar_Click(sender As Object, e As EventArgs) Handles
btnListar.Click
        ListBox1.Items.Clear()
        Dim archivos() As String = Directory.GetFiles("C:\")
        For Each archivo In archivos
            ListBox1.Items.Add(archivo)
        Next
    End Sub
End Class
```

Programa 10 – Listar subdirectorios

Enunciado: Muestra todos los subdirectorios de una carpeta.
Explicación: `Directory.GetDirectories` devuelve un arreglo con los nombres de carpetas.

```
Imports System.IO

Public Class Form1
    Private Sub btnSubDirs_Click(sender As Object, e As EventArgs) Handles
btnSubDirs.Click
        ListBox1.Items.Clear()
        Dim dirs() As String = Directory.GetDirectories("C:\")
        For Each dir In dirs
            ListBox1.Items.Add(dir)
        Next
    End Sub
End Class
```

Programa 11 – Leer archivo binario

Enunciado: Permite leer datos binarios de un archivo `.bin`.
Explicación: Se usa `BinaryReader` para leer valores en formato binario.

```
Imports System.IO

Public Class Form1
    Private Sub btnLeerBin_Click(sender As Object, e As EventArgs) Handles
btnLeerBin.Click
        Using br As New BinaryReader(File.Open("C:\datos.bin", FileMode.Open))
            Dim valor As Integer = br.ReadInt32()
            MessageBox.Show("Valor leído: " & valor)
        End Using
    End Sub
End Class
```

Programa 12 – Escribir archivo binario

Enunciado: Permite crear y escribir datos binarios en un archivo `.bin`.

Explicación: `BinaryWriter` escribe enteros, cadenas u otros datos en formato binario.

```
Imports System.IO

Public Class Form1
    Private Sub btnEscribirBin_Click(sender As Object, e As EventArgs) Handles
btnEscribirBin.Click
        Using bw As New BinaryWriter(File.Open("C:\datos.bin", FileMode.Create))
            bw.Write(12345)
        End Using
        MessageBox.Show("Archivo binario creado.")
    End Sub
End Class
```

Programa 13 – Copiar archivo binario

Enunciado: Copia un archivo binario a otra ubicación.

Explicación: `File.Copy` funciona igual para archivos binarios.

```
Imports System.IO

Public Class Form1
    Private Sub btnCopiarBin_Click(sender As Object, e As EventArgs) Handles
btnCopiarBin.Click
        File.Copy("C:\datos.bin", "D:\datos_copia.bin", True)
        MessageBox.Show("Archivo binario copiado.")
    End Sub
End Class
```

Programa 14 – Contar líneas de un archivo de texto

Enunciado: Cuenta cuántas líneas tiene un archivo de texto.

Explicación: `File.ReadAllLines` devuelve un arreglo de líneas.

```
Imports System.IO

Public Class Form1
    Private Sub btnContarLineas_Click(sender As Object, e As EventArgs) Handles
btnContarLineas.Click
        Dim lineas() As String = File.ReadAllLines("C:\archivo1.txt")
        MessageBox.Show("Número de líneas: " & lineas.Length)
    End Sub
End Class
```

Programa 15 – Buscar texto en un archivo

Enunciado: Permite buscar un texto específico dentro de un archivo.

Explicación: Recorremos cada línea con `StreamReader` y verificamos coincidencias.

```
Imports System.IO

Public Class Form1
    Private Sub btnBuscar_Click(sender As Object, e As EventArgs) Handles
btnBuscar.Click
        Dim textoBuscar As String = txtBuscar.Text
        Using sr As New StreamReader("C:\archivo1.txt")
            Dim encontrado As Boolean = False
            While Not sr.EndOfStream
                If sr.ReadLine().Contains(textoBuscar) Then
                    encontrado = True
                    Exit While
                End If
            End While
            MessageBox.Show(If(encontrado, "Texto encontrado", "Texto no
encontrado"))
        End Using
    End Sub
End Class
```

Programa 16 – Guardar múltiples líneas en un archivo

Enunciado: Permite al usuario guardar varias líneas desde un ListBox en un archivo de texto.

Explicación: Iteramos `ListBox.Items` y escribimos con `StreamWriter`.

```
Imports System.IO

Public Class Form1
    Private Sub btnGuardarLineas_Click(sender As Object, e As EventArgs) Handles
btnGuardarLineas.Click
        Using sw As New StreamWriter("C:\archivo_multilinea.txt")
            For Each linea In ListBox1.Items
                sw.WriteLine(linea)
            Next
        End Using
        MessageBox.Show("Archivo guardado con múltiples líneas.")
    End Sub
End Class
```

Programa 17 – Leer archivo línea por línea

Enunciado: Muestra el contenido de un archivo línea por línea en un ListBox.
Explicación: Usamos `StreamReader.ReadLine` dentro de un bucle.

```
Imports System.IO

Public Class Form1
    Private Sub btnLeerLineas_Click(sender As Object, e As EventArgs) Handles
btnLeerLineas.Click
        ListBox1.Items.Clear()
        Using sr As New StreamReader("C:\archivo_multilinea.txt")
            While Not sr.EndOfStream
                ListBox1.Items.Add(sr.ReadLine())
            End While
        End Using
    End Sub
End Class
```

Programa 18 – Crear archivo si no existe

Enunciado: Crea un archivo solo si no existe para evitar sobrescribirlo.
Explicación: `File.Exists` verifica la existencia del archivo.

```
Imports System.IO

Public Class Form1
    Private Sub btnCrearSiNoExiste_Click(sender As Object, e As EventArgs)
Handles btnCrearSiNoExiste.Click
        Dim ruta As String = "C:\archivo_nuevo.txt"
        If Not File.Exists(ruta) Then
            File.Create(ruta).Close()
            MessageBox.Show("Archivo creado.")
        Else
            MessageBox.Show("El archivo ya existe.")
        End If
    End Sub
End Class
```

Programa 19 – Obtener propiedades de un archivo

Enunciado: Muestra el tamaño y la fecha de creación de un archivo.
Explicación: `FileInfo` permite acceder a la información del archivo.

```
Imports System.IO

Public Class Form1
    Private Sub btnPropiedades_Click(sender As Object, e As EventArgs) Handles
btnPropiedades.Click
        Dim fi As New FileInfo("C:\archivo1.txt")
        MessageBox.Show("Tamaño: " & fi.Length & " bytes" & vbCrLf & "Creado: "
& fi.CreationTime)
    End Sub
End Class
```

Programa 20 – Renombrar un archivo

Enunciado: Permite cambiar el nombre de un archivo existente.

Explicación: `File.Move` puede renombrar un archivo moviéndolo con otro nombre.

```
Imports System.IO

Public Class Form1
    Private Sub btnRenombrar_Click(sender As Object, e As EventArgs) Handles
btnRenombrar.Click
        File.Move("C:\archivo1.txt", "C:\archivo_renombrado.txt")
        MessageBox.Show("Archivo renombrado.")
    End Sub
End Class
```

Ejercicios

22.1 Try, Catch, Finally

Programa 1: Captura de división por cero

Enunciado: Pide al usuario dos números y realiza la división; captura el error si el divisor es cero.

```
Public Class Form1
    Private Sub btnDividir_Click(sender As Object, e As EventArgs) Handles
btnDividir.Click
        Try
            Dim a As Integer = CInt(txtNumero1.Text)
            Dim b As Integer = CInt(txtNumero2.Text)
            Dim resultado As Double = a / b
            MessageBox.Show("Resultado: " & resultado)
        Catch ex As DivideByZeroException
            MessageBox.Show("Error: No se puede dividir entre cero.")
        Catch ex As Exception
            MessageBox.Show("Error: " & ex.Message)
        Finally
            MessageBox.Show("Operación finalizada.")
        End Try
    End Sub
End Class
```

Programa 2: Conversión de texto a número

Enunciado: Intenta convertir un texto en número; captura errores si el texto no es numérico.

```
Public Class Form2
    Private Sub btnConvertir_Click(sender As Object, e As EventArgs) Handles
btnConvertir.Click
        Try
            Dim numero As Integer = CInt(txtInput.Text)
            MessageBox.Show("Número convertido: " & numero)
        Catch ex As FormatException
            MessageBox.Show("Error: Entrada no es un número válido.")
        Finally
            txtInput.Clear()
        End Try
    End Sub
End Class
```

Programa 3: Manejo de múltiples excepciones

Enunciado: Suma dos números, pero captura tanto errores de conversión como división por cero.

```
Public Class Form3
    Private Sub btnCalcular_Click(sender As Object, e As EventArgs) Handles
btnCalcular.Click
        Try
            Dim a As Integer = CInt(txtA.Text)
            Dim b As Integer = CInt(txtB.Text)
            Dim resultado As Double = a / b
            MessageBox.Show("Resultado: " & resultado)
        Catch ex As DivideByZeroException
            MessageBox.Show("No se puede dividir entre cero.")
        Catch ex As FormatException
            MessageBox.Show("Debe ingresar números válidos.")
        Finally
            txtA.Clear()
            txtB.Clear()
        End Try
    End Sub
End Class
```

Programa 4: Manejo de excepciones genéricas

Enunciado: Intenta abrir un archivo; captura cualquier error inesperado.

```
Imports System.IO
Public Class Form4
    Private Sub btnAbrirArchivo_Click(sender As Object, e As EventArgs) Handles
btnAbrirArchivo.Click
        Try
            Dim contenido As String = File.ReadAllText("C:\archivo.txt")
            MessageBox.Show(contenido)
        Catch ex As Exception
            MessageBox.Show("Ocurrió un error: " & ex.Message)
        Finally
            MessageBox.Show("Intento de lectura finalizado.")
        End Try
    End Sub
End Class
```

Programa 5: Finally para cerrar recursos

Enunciado: Lee un archivo usando StreamReader y asegura el cierre del recurso en Finally.

```
Imports System.IO
Public Class Form5
    Private Sub btnLeer_Click(sender As Object, e As EventArgs) Handles
btnLeer.Click
        Dim lector As StreamReader = Nothing
        Try
            lector = New StreamReader("C:\archivo.txt")
            Dim linea As String = lector.ReadLine()
            MessageBox.Show(linea)
        Catch ex As Exception
            MessageBox.Show("Error al leer archivo: " & ex.Message)
        Finally
            If Not lector Is Nothing Then lector.Close()
        End Try
    End Sub
End Class
```

22.2 Excepciones personalizadas

Programa 6: Excepción personalizada simple

Enunciado: Define una excepción personalizada para números negativos.

```
Public Class NumeroNegativoException
    Inherits Exception
    Public Sub New()
        MyBase.New("No se permiten números negativos.")
    End Sub
End Class

Public Class Form6
    Private Sub btnValidar_Click(sender As Object, e As EventArgs) Handles
btnValidar.Click
        Try
            Dim num As Integer = CInt(txtNumero.Text)
            If num < 0 Then
                Throw New NumeroNegativoException()
            End If
            MessageBox.Show("Número válido: " & num)
        Catch ex As NumeroNegativoException
            MessageBox.Show(ex.Message)
        End Try
    End Sub
End Class
```

Programa 7: Validación de edad

Enunciado: Lanza una excepción personalizada si el usuario es menor de edad.

```
Public Class MenorDeEdadException
    Inherits Exception
    Public Sub New()
        MyBase.New("Usuario menor de edad, acceso denegado.")
    End Sub
End Class

Public Class Form7
    Private Sub btnValidarEdad_Click(sender As Object, e As EventArgs) Handles
btnValidarEdad.Click
        Try
            Dim edad As Integer = CInt(txtEdad.Text)
            If edad < 18 Then
                Throw New MenorDeEdadException()
            End If
            MessageBox.Show("Edad válida.")
        Catch ex As MenorDeEdadException
            MessageBox.Show(ex.Message)
        End Try
    End Sub
End Class
```

Programa 8: Salario negativo

Enunciado: Detecta salario negativo con excepción personalizada.

```
Public Class SalarioNegativoException
    Inherits Exception
    Public Sub New()
        MyBase.New("El salario no puede ser negativo.")
    End Sub
End Class

Public Class Form8
    Private Sub btnCalcularSalario_Click(sender As Object, e As EventArgs)
Handles btnCalcularSalario.Click
        Try
            Dim salario As Double = CDbl(txtSalario.Text)
            If salario < 0 Then Throw New SalarioNegativoException()
            MessageBox.Show("Salario registrado: " & salario)
        Catch ex As SalarioNegativoException
            MessageBox.Show(ex.Message)
        End Try
    End Sub
End Class
```

Programa 9: Excepción personalizada para notas

Enunciado: Lanza excepción si la nota ingresada está fuera del rango 0-10.

```
Public Class NotaFueraDeRangoException
    Inherits Exception
    Public Sub New()
        MyBase.New("La nota debe estar entre 0 y 10.")
    End Sub
End Class

Public Class Form9
    Private Sub btnValidarNota_Click(sender As Object, e As EventArgs) Handles
btnValidarNota.Click
        Try
            Dim nota As Double = CDbl(txtNota.Text)
            If nota < 0 OrElse nota > 10 Then
                Throw New NotaFueraDeRangoException()
            End If
            MessageBox.Show("Nota válida: " & nota)
        Catch ex As NotaFueraDeRangoException
            MessageBox.Show(ex.Message)
        End Try
    End Sub
End Class
```

Programa 10: Excepción personalizada con parámetros

Enunciado: Personaliza el mensaje de error según el valor ingresado.

```
Public Class ValorInvalidoException
    Inherits Exception
    Public Sub New(valor As String)
        MyBase.New("El valor '" & valor & "' es inválido.")
    End Sub
End Class

Public Class Form10
    Private Sub btnValidar_Click(sender As Object, e As EventArgs) Handles
btnValidar.Click
        Try
            Dim valor As String = txtValor.Text
            If valor = "" Then Throw New ValorInvalidoException(valor)
            MessageBox.Show("Valor aceptado: " & valor)
        Catch ex As ValorInvalidoException
            MessageBox.Show(ex.Message)
        End Try
    End Sub
End Class
```

22.3 Herramientas de depuración en Visual Studio

Programa 11: Debug.WriteLine

Enunciado: Usa depuración para mostrar datos sin interrumpir la ejecución.

```
Imports System.Diagnostics
Public Class Form11
    Private Sub btnDebug_Click(sender As Object, e As EventArgs) Handles
btnDebug.Click
        Dim valor As Integer = 10
        Debug.WriteLine("Valor inicial: " & valor)
        valor += 5
        Debug.WriteLine("Valor final: " & valor)
        MessageBox.Show("Revisa la ventana de salida para depuración.")
    End Sub
End Class
```

Programa 12: Uso de Breakpoint y Watch

Enunciado: Permite al usuario cambiar un número y mirar su valor en depuración.

```
Public Class Form12
    Private Sub btnProbarBreakpoint_Click(sender As Object, e As EventArgs)
Handles btnProbarBreakpoint.Click
        Dim numero As Integer = CInt(txtNumero.Text) ' Colocar breakpoint aquí
        numero += 1 ' Observar en Watch
        MessageBox.Show("Número incrementado: " & numero)
    End Sub
End Class
```

Programa 13: Seguimiento de errores con Try-Catch

Enunciado: Simula un error y revisa la traza en depuración.

```
Public Class Form13
    Private Sub btnSimularError_Click(sender As Object, e As EventArgs) Handles
btnSimularError.Click
        Try
            Dim arreglo(2) As Integer
            arreglo(5) = 10
        Catch ex As Exception
            Debug.WriteLine("Error detectado: " & ex.Message)
            MessageBox.Show("Error registrado en depuración.")
        End Try
    End Sub
End Class
```

Programa 14: Uso de Debug.Assert

Enunciado: Verifica condiciones en tiempo de depuración.

```
Imports System.Diagnostics
Public Class Form14
    Private Sub btnAssert_Click(sender As Object, e As EventArgs) Handles
btnAssert.Click
        Dim valor As Integer = CInt(txtNumero.Text)
        Debug.Assert(valor > 0, "El valor debe ser mayor que cero")
        MessageBox.Show("Verifica la ventana de salida para posibles fallos.")
    End Sub
End Class
```

Programa 15: Depuración de bucles

Enunciado: Observa valores dentro de un bucle usando Debug.

```
Imports System.Diagnostics
Public Class Form15
    Private Sub btnBucle_Click(sender As Object, e As EventArgs) Handles
btnBucle.Click
        For i As Integer = 1 To 5
            Debug.WriteLine("Iteración: " & i)
        Next
        MessageBox.Show("Revisa la ventana de salida.")
    End Sub
End Class
```

Programa 16: Trace.WriteLine

Enunciado: Usa Trace para seguimiento en ejecución y producción.

```
Imports System.Diagnostics
Public Class Form16
    Private Sub btnTrace_Click(sender As Object, e As EventArgs) Handles
btnTrace.Click
        Trace.WriteLine("Programa iniciado.")
        Trace.WriteLine("Operación completada.")
        MessageBox.Show("Verifica ventana de salida para Trace.")
    End Sub
End Class
```

Programa 17: Depuración de formularios múltiples

Enunciado: Cambia entre formularios y registra eventos.

```
Imports System.Diagnostics
Public Class Form17
    Private Sub btnAbrirForm2_Click(sender As Object, e As EventArgs) Handles
btnAbrirForm2.Click
        Debug.WriteLine("Abriendo Form2")
        Dim f2 As New Form2()
        f2.Show()
    End Sub
End Class
```

Programa 18: Depuración de eventos de botón

Enunciado: Rastrea cuántas veces se pulsa un botón.

```
Imports System.Diagnostics
Public Class Form18
    Private contador As Integer = 0
    Private Sub btnContar_Click(sender As Object, e As EventArgs) Handles
btnContar.Click
        contador += 1
        Debug.WriteLine("Botón pulsado: " & contador & " veces")
        MessageBox.Show("Botón pulsado " & contador & " veces")
    End Sub
End Class
```

Programa 19: Depuración de cambios de texto

Enunciado: Muestra depuración al cambiar el texto de un TextBox.

```
Imports System.Diagnostics
Public Class Form19
    Private Sub txtMonitor_TextChanged(sender As Object, e As EventArgs) Handles
txtMonitor.TextChanged
        Debug.WriteLine("Texto cambiado: " & txtMonitor.Text)
    End Sub
End Class
```

Programa 20: Combinación de Try-Catch y Debug

Enunciado: Captura un error y lo registra en depuración.

```
Imports System.Diagnostics
Public Class Form20
    Private Sub btnCombinado_Click(sender As Object, e As EventArgs) Handles
btnCombinado.Click
        Try
            Dim x As Integer = CInt(txtValor.Text)
            Dim y As Integer = CInt(txtDivisor.Text)
            Dim resultado As Double = x / y
            MessageBox.Show("Resultado: " & resultado)
        Catch ex As Exception
            Debug.WriteLine("Error: " & ex.Message)
            MessageBox.Show("Se produjo un error, revisa depuración.")
        End Try
    End Sub
End Class
```

Ejercicios

23.1 Calculadora con TextBox, Label y Button

Enunciado: Programa que realiza operaciones básicas (+, -, *, /) usando botones para cada operación y muestra el resultado en un Label.

```
Public Class Form1
    Private Sub btnSumar_Click(sender As Object, e As EventArgs) Handles
btnSumar.Click
        Try
            Dim num1 As Double = Double.Parse(txtNumero1.Text)
            Dim num2 As Double = Double.Parse(txtNumero2.Text)
            lblResultado.Text = "Resultado: " & (num1 + num2).ToString()
        Catch ex As Exception
            MessageBox.Show("Ingrese números válidos")
        End Try
    End Sub
End Class
```

23.2 Agenda de contactos con interacción entre formularios

Enunciado: Permite agregar contactos desde un formulario principal y abrir un formulario secundario para mostrar la lista de contactos.

```
Public Class Form1
    Private contactos As New List(Of String)
    Private Sub btnAgregar_Click(sender As Object, e As EventArgs) Handles
btnAgregar.Click
        contactos.Add(txtNombre.Text)
        txtNombre.Clear()
    End Sub
    Private Sub btnVerContactos_Click(sender As Object, e As EventArgs) Handles
btnVerContactos.Click
        Dim frmLista As New Form2(contactos)
        frmLista.Show()
    End Sub
End Class

Public Class Form2
    Public Sub New(lista As List(Of String))
        InitializeComponent()
        For Each c In lista
            ListBox1.Items.Add(c)
        Next
    End Sub
End Class
```

23.3 Aplicación de inventario con DataGridView

Enunciado: Muestra productos en un `DataGridView` y permite agregar filas con nombre y cantidad.

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        DataGridView1.ColumnCount = 2
        DataGridView1.Columns(0).Name = "Producto"
        DataGridView1.Columns(1).Name = "Cantidad"
    End Sub
    Private Sub btnAgregar_Click(sender As Object, e As EventArgs) Handles
btnAgregar.Click
        DataGridView1.Rows.Add(txtProducto.Text, txtCantidad.Text)
    End Sub
End Class
```

23.4 Bloc de notas con menús y ToolStrip

Enunciado: Bloc de notas básico con menú para abrir, guardar y limpiar texto usando `MenuStrip` y `ToolStrip`.

```
Public Class Form1
    Private Sub AbrirToolStripMenuItem_Click(sender As Object, e As EventArgs)
Handles AbrirToolStripMenuItem.Click
        Dim openFile As New OpenFileDialog
        If openFile.ShowDialog() = DialogResult.OK Then
            txtNotas.Text = IO.File.ReadAllText(openFile.FileName)
        End If
    End Sub
    Private Sub GuardarToolStripMenuItem_Click(sender As Object, e As EventArgs)
Handles GuardarToolStripMenuItem.Click
        Dim saveFile As New SaveFileDialog
        If saveFile.ShowDialog() = DialogResult.OK Then
            IO.File.WriteAllText(saveFile.FileName, txtNotas.Text)
        End If
    End Sub
    Private Sub LimpiarToolStripMenuItem_Click(sender As Object, e As EventArgs)
Handles LimpiarToolStripMenuItem.Click
        txtNotas.Clear()
    End Sub
End Class
```

23.5 Reproductor multimedia con botones y diálogos

Enunciado: Permite cargar un archivo de audio y reproducirlo, pausar o detener.

```
Imports System.Media

Public Class Form1
    Private player As SoundPlayer
    Private Sub btnCargar_Click(sender As Object, e As EventArgs) Handles
btnCargar.Click
        Dim openFile As New OpenFileDialog
        openFile.Filter = "Archivos WAV|*.wav"
        If openFile.ShowDialog() = DialogResult.OK Then
            player = New SoundPlayer(openFile.FileName)
        End If
    End Sub
    Private Sub btnReproducir_Click(sender As Object, e As EventArgs) Handles
btnReproducir.Click
        If player IsNot Nothing Then player.Play()
    End Sub
    Private Sub btnDetener_Click(sender As Object, e As EventArgs) Handles
btnDetener.Click
        If player IsNot Nothing Then player.Stop()
    End Sub
End Class
```

23.6 Contador con Timer

Enunciado: Programa que cuenta segundos automáticamente usando un `Timer` y muestra el valor en un `Label`.

```
Public Class Form1

    Private contador As Integer = 0

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

        Timer1.Interval = 1000

        Timer1.Start()

    End Sub

    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles
Timer1.Tick

        contador += 1

        lblContador.Text = "Segundos: " & contador

    End Sub

End Class
```

23.7 Juego de adivinar número

Enunciado: Genera un número aleatorio y permite que el usuario adivine con indicaciones "Mayor" o "Menor".

```
Public Class Form1

    Private numero As Integer

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

        Dim rnd As New Random()

        numero = rnd.Next(1, 101)

    End Sub

    Private Sub btnAdivinar_Click(sender As Object, e As EventArgs) Handles
btnAdivinar.Click

        Dim intento As Integer = CInt(txtIntento.Text)

        If intento = numero Then

            MessageBox.Show("¡Correcto!")

        ElseIf intento < numero Then

            MessageBox.Show("Mayor")

        Else

            MessageBox.Show("Menor")

        End If

    End Sub

End Class
```

23.8 Conversor de temperatura

Enunciado: Convierte grados Celsius a Fahrenheit y Kelvin usando TextBox y Button.

```
Public Class Form1

    Private Sub btnConvertir_Click(sender As Object, e As EventArgs) Handles
btnConvertir.Click

        Dim c As Double = CDbl(txtCelsius.Text)

        txtFahrenheit.Text = (c * 9 / 5 + 32).ToString()

        txtKelvin.Text = (c + 273.15).ToString()

    End Sub

End Class
```

23.9 Lista desplegable de colores

Enunciado: Cambia el color de fondo del formulario según la selección de un `ComboBox`.

```
Public Class Form1

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

        ComboBox1.Items.AddRange(New String() {"Rojo", "Verde", "Azul"})

    End Sub

    Private Sub ComboBox1_SelectedIndexChanged(sender As Object, e As EventArgs)
Handles ComboBox1.SelectedIndexChanged

        Select Case ComboBox1.SelectedItem.ToString()

            Case "Rojo" : Me.BackColor = Color.Red

            Case "Verde" : Me.BackColor = Color.Green

            Case "Azul" : Me.BackColor = Color.Blue

        End Select

    End Sub

End Class
```

23.10 Registro de notas con ListBox

Enunciado: Permite agregar y eliminar notas en un `ListBox`.

```
Public Class Form1

    Private Sub btnAgregar_Click(sender As Object, e As EventArgs) Handles
btnAgregar.Click

        ListBox1.Items.Add(txtNota.Text)

        txtNota.Clear()

    End Sub

    Private Sub btnEliminar_Click(sender As Object, e As EventArgs) Handles
btnEliminar.Click

        If ListBox1.SelectedIndex >= 0 Then
ListBox1.Items.RemoveAt(ListBox1.SelectedIndex)

    End Sub

End Class
```

23.11 Reloj en tiempo real

Enunciado: Muestra la hora actual actualizada cada segundo en un Label.

```
Public Class Form1

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

        Timer1.Interval = 1000

        Timer1.Start()

    End Sub

    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles
Timer1.Tick

        lblReloj.Text = DateTime.Now.ToString("HH:mm:ss")

    End Sub

End Class
```

23.12 Calculadora de promedio

Enunciado: Calcula el promedio de varios números ingresados en TextBox y muestra el resultado.

```
Public Class Form1

    Private Sub btnCalcular_Click(sender As Object, e As EventArgs) Handles
btnCalcular.Click

        Dim numeros() As String = txtNumeros.Text.Split(","c)

        Dim suma As Double = 0

        For Each n In numeros

            suma += CDbl(n)

        Next

        lblPromedio.Text = "Promedio: " & (suma / numeros.Length)

    End Sub

End Class
```

23.13 Temporizador con ProgressBar

Enunciado: Incrementa una barra de progreso cada segundo hasta completarse.

```
Public Class Form1

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

        ProgressBar1.Maximum = 10

        Timer1.Interval = 1000

        Timer1.Start()

    End Sub

    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles
Timer1.Tick

        If ProgressBar1.Value < ProgressBar1.Maximum Then

            ProgressBar1.Value += 1

        Else

            Timer1.Stop()

            MessageBox.Show("¡Tiempo completado!")

        End If

    End Sub

End Class
```

23.14 Simulador de cajero

Enunciado: Permite depositar y retirar dinero mostrando el saldo actual.

```
Public Class Form1

    Private saldo As Double = 0

    Private Sub btnDepositar_Click(sender As Object, e As EventArgs) Handles
btnDepositar.Click

        saldo += CDbl(txtCantidad.Text)

        lblSaldo.Text = "Saldo: " & saldo

    End Sub

    Private Sub btnRetirar_Click(sender As Object, e As EventArgs) Handles
btnRetirar.Click

        Dim retiro As Double = CDbl(txtCantidad.Text)

        If retiro <= saldo Then saldo -= retiro

        lblSaldo.Text = "Saldo: " & saldo

    End Sub

End Class
```

23.15 Generador de contraseña aleatoria

Enunciado: Crea contraseñas aleatorias de 8 caracteres usando letras y números.

```
Public Class Form1

    Private Sub btnGenerar_Click(sender As Object, e As EventArgs) Handles
btnGenerar.Click

        Dim caracteres As String =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789"

        Dim rnd As New Random()

        Dim contraseña As String = ""

        For i = 1 To 8

            contraseña &= caracteres(rnd.Next(caracteres.Length))

        Next

        lblContraseña.Text = contraseña

    End Sub

End Class
```

23.16 Agenda avanzada con interacción entre formularios

Enunciado: Permite agregar contactos en un formulario y mostrarlos en un `DataGridView` de otro formulario.

```
' Form1 - Agregar contactos
Public Class Form1
    Private Sub btnAgregar_Click(sender As Object, e As EventArgs) Handles
btnAgregar.Click
        Dim contacto As String = txtNombre.Text & " - " & txtTelefono.Text
        Form2.AgregarContacto(contacto)
        txtNombre.Clear()
        txtTelefono.Clear()
    End Sub
End Class

' Form2 - Mostrar contactos
Public Class Form2
    Public Sub AgregarContacto(contacto As String)
        DataGridView1.Rows.Add(contacto)
    End Sub
End Class
```

23.17 Mini-editor gráfico

Enunciado: Permite dibujar líneas en un `PictureBox` usando el mouse.

```
Public Class Form1
    Private inicio As Point
    Private dibujar As Boolean = False
    Private Sub PictureBox1_MouseDown(sender As Object, e As MouseEventArgs)
Handles PictureBox1.MouseDown
        dibujar = True
        inicio = e.Location
    End Sub
    Private Sub PictureBox1_MouseMove(sender As Object, e As MouseEventArgs)
Handles PictureBox1.MouseMove
        If dibujar Then
            Dim g As Graphics = PictureBox1.CreateGraphics()
            g.DrawLine(Pens.Black, inicio, e.Location)
            inicio = e.Location
        End If
    End Sub
    Private Sub PictureBox1_MouseUp(sender As Object, e As MouseEventArgs)
Handles PictureBox1.MouseUp
        dibujar = False
    End Sub
End Class
```

23.18 Lista de tareas con CheckBox

Enunciado: Permite agregar tareas a un `ListBox` y marcarlas como completadas usando `CheckedListBox`.

```
Public Class Form1
    Private Sub btnAgregar_Click(sender As Object, e As EventArgs) Handles
btnAgregar.Click
        CheckedListBox1.Items.Add(txtTarea.Text, False)
        txtTarea.Clear()
    End Sub
    Private Sub btnEliminar_Click(sender As Object, e As EventArgs) Handles
btnEliminar.Click
        For i As Integer = CheckedListBox1.Items.Count - 1 To 0 Step -1
            If CheckedListBox1.GetItemChecked(i) Then
CheckedListBox1.Items.RemoveAt(i)
        Next
    End Sub
End Class
```

23.19 CRUD básico con DataGridView

Enunciado: Permite agregar, editar y eliminar registros en un `DataGridView` dentro del mismo formulario.

```
Public Class Form1
    Private Sub btnAgregar_Click(sender As Object, e As EventArgs) Handles
btnAgregar.Click
        DataGridView1.Rows.Add(txtID.Text, txtNombre.Text)
        txtID.Clear()
        txtNombre.Clear()
    End Sub
    Private Sub btnEliminar_Click(sender As Object, e As EventArgs) Handles
btnEliminar.Click
        For Each row As DataGridViewRow In DataGridView1.SelectedRows
            DataGridView1.Rows.Remove(row)
        Next
    End Sub
    Private Sub btnEditar_Click(sender As Object, e As EventArgs) Handles
btnEditar.Click
        If DataGridView1.SelectedRows.Count > 0 Then
            Dim row As DataGridViewRow = DataGridView1.SelectedRows(0)
            row.Cells(0).Value = txtID.Text
            row.Cells(1).Value = txtNombre.Text
        End If
    End Sub
End Class
```

23.20 Reproductor de sonidos

Enunciado: Reproduce un archivo de audio seleccionado por el usuario con botones de Play y Stop.

```
Imports System.Media
Public Class Form1
    Private reproductor As New SoundPlayer()
    Private Sub btnCargar_Click(sender As Object, e As EventArgs) Handles
btnCargar.Click
        OpenFileDialog1.Filter = "Archivos WAV|*.wav"
        If OpenFileDialog1.ShowDialog() = DialogResult.OK Then
            reproductor.SoundLocation = OpenFileDialog1.FileName
        End If
    End Sub
    Private Sub btnPlay_Click(sender As Object, e As EventArgs) Handles
btnPlay.Click
        reproductor.Play()
    End Sub
    Private Sub btnStop_Click(sender As Object, e As EventArgs) Handles
btnStop.Click
        reproductor.Stop()
    End Sub
End Class
```

Epílogo

Llegamos al final de este recorrido por el mundo de Visual Basic .NET enfocado en formularios. A lo largo de estas páginas, hemos explorado desde los conceptos más básicos del lenguaje hasta técnicas avanzadas para crear aplicaciones completas y funcionales, integrando formularios, controles, eventos y acceso a datos. Mi objetivo ha sido brindarte no solo el conocimiento técnico, sino también la confianza para que puedas aplicar lo aprendido en tus propios proyectos y desafíos de programación.

Programar en Visual Basic .NET es mucho más que escribir líneas de código: es diseñar soluciones, dar vida a ideas y aprender a resolver problemas de manera creativa y eficiente. Espero que este libro te haya proporcionado las herramientas necesarias para seguir creciendo como desarrollador, y que cada ejemplo y ejercicio te haya acercado un paso más a la maestría en la creación de aplicaciones con formularios.

Quiero cerrar nuevamente agradeciendo a mi esposa, **Marisa**, por su apoyo constante durante la creación de este libro. Su paciencia, aliento y confianza fueron fundamentales para que pudiera materializar este proyecto. Cada página escrita lleva también un pedacito de su dedicación y cariño.

Finalmente, deseo que este libro sea solo el comienzo de tu viaje en la programación con Visual Basic .NET, y que sigas explorando, creando y aprendiendo, siempre con pasión y curiosidad. Recuerda que cada línea de código es una oportunidad para construir algo nuevo, útil y personal.